Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Žiga Vučko

Napovedovanje kategorij novičarskih člankov s pomočjo meta-podatkov s spleta

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Lovro Šubelj

Ljubljana, 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Event Registry je sistem Instituta Jožef Stefan, ki v realnem času zbira spletne novičarske članke, v njih poskuša prepoznati dogodke in jih uvrstiti v kategorije. Pri tem pa problem uvrščanja še ni zadovoljivo rešen. V diplomskem delu preizkusite pristope strojnega učenja in odkrivanja znanj iz besedil za namene avtomatskega uvrščanja člankov v kategorije ter pa pristope aktivnega učenja za namene pohitritve ročnega označevanja člankov. Posebno pozornost namenite izboljšavi uvrščanja z uporabo meta-podatkov iz spletnih virov kot je Never-Ending Language Learner, ki ga razvijajo na Univerzi Carnegie Mellon.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Žiga Vučko sem avtor diplomskega dela z naslovom:

Napovedovanje kategorij novičarskih člankov s pomočjo meta-podatkov s spleta

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Lovra Šublja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 13. septembra 2015 Podpis avtorja:

Za vso strokovno podporo in pomoč pri izdelavi diplomskega dela bi se zahvalil doc. dr. Lovru Šublju, as. dr. Aljažu Košmerlju iz Instituta Jožef Stefan in as. dr. Slavku Žitniku. Hkrati bi se zahvalil Gregorju Lebanu, Marku Grobelniku in Maji Škrjanc ter ostalim članom Laboratorija za umetno inteligenco Instituta Jožef Stefan, ki so mi omogočili dostop do široke palete kakovostnih podatkov, ki sem jih s pridom uporabil pri izdelavi diplomskega dela.

Za vso moralno in finančno podporo, ki sta mi jo nudila tekom let mojega šolanja, se moram zahvaliti materi Poloni in očetu Darku. Hkrati se zahvaljujem babici Vidi za vse prijazne besede in sladke dobrote, ki mi jih je prinašala med dolgimi urami pisanja. Posebno zahvalo pa bi namenil tudi Tini ⋆, ki je prenašala moja razpoloženjska nihanja med pisanjem diplome in me spodbujala pri delu.

Na koncu bi se zahvalil še prijatelju Kristijanu za dolge konstruktivne pogovore o problemih diplomskega dela.

"You know nothing, Jon Snow."

— Ygritte

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Strojno učenje **5**

2.1 Nadzorovano učenje 7

2.1.1 Klasifikacija 8

2.1.1.1 Logistična regresija 10

2.1.1.2 Metoda podpornih vektorjev 12

2.1.1.3 Naključni gozdovi 16

2.1.1.4 Metoda k-najbližjih sosedov 17

2.1.2 Validacija in merjenje uspešnosti klasifikacije 20

2.2 Aktivno učenje 24

2.3 Odkrivanje znanj iz besedil 27

3 Uporabljene tehnologije in orodja **31**

3.1 Enrycher, IJS Newsfeed in Event Registry 31

3.2 Portal dogodkov Wikipedia 35

3.3 Guardian API 36

3.4 Sistem NELL 37

3.5 Programska orodja 42

4 Napovedovanje kategorij novičarskih člankov **43**
4.1 Članki in meta-podatki 43
4.2 Klasifikacijski modeli 46
4.3 Pristopi k aktivnemu učenju 48
4.4 Aktivno učenje na neoznačenih člankih 51

5 Rezultati in diskusija **53**
5.1 Klasifikacijski modeli 53
5.2 Pristopi k aktivnemu učenju 61
5.3 Aktivno učenje na neoznačenih člankih 64

6 Sklepne ugotovitve **65**

A Programska koda **69**

Literatura **79**

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application Programming Interface	aplikacijski programski vmesnik
CA	Classification Accuracy	klasifikacijska točnost
DMoz	Directory Mozilla	ročno kuriran večjezični imenik spletnih povezav
EM	Expectation-Maximization	pričakovanje-maksimizacija
FN	False Negatives	napačno klasificirani negativni primeri
FP	False Positives	napačno klasificirani pozitivni primeri
HTML	Hyper Text Markup Language	označevalni jezik spletnih strani
IJS	Jožef Stefan Institute	Institut Jožef Stefan
JSON	JavaScript Object Notation	objektni zapis JavaScript
k**-NN**	k-Nearest Neighbors	k-najbližjih sosedov
NELL	Never-Ending Language Learner	sistem, ki se neprekinjeno uči s spleta
PHP	PHP Hypertext Preprocessor	programski jezik za dinamične spletne strani
RDF	Resource Description Framework	ogrodje/sistem za opisovanje virov
RSS	Rich Site Summary	družina datotečnih oblik XML za spletno zlaganje
SVM	Support Vector Machine	metoda podpornih vektorjev

kratica	**angleško**	**slovensko**
TF-IDF	Term Frequency - Inverse Document Frequecy	frekvenca besede v dokumentu - relativna redkost besede v korpusu
TN	True Negatives	pravilno klasificirani negativni primeri
TP	True Positives	pravilno klasificirani pozitivni primeri
URL	Uniform Resource Locator	enolični krajevnik vira
XML	Extensible Markup Language	razširljiv označevalni jezik

Povzetek

Področje strojnega učenja, ki se ukvarja z odkrivanjem znanj iz besedil se dandanes razvija z izjemno hitrostjo in kot tako ponuja številne priložnosti. Na tem področju deluje tudi Laboratorij za umetno inteligenco Instituta Jožef Stefan, kjer razvijajo sistem Event Registry, ki v realnem času zbira spletne novičarske članke, jih združuje v dogodke in iz njih ekstrahira pomembne informacije. Komponenta sistema, ki uvršča članke v kategorije še ni dodelana. Kot odgovor na to, smo se v diplomskem delu lotili nadgradnje referenčnega modela. Rezultati našega dela so bili pozitivni, saj smo izboljšali napovedno točnost klasifikacije poljubnih novičarskih člankov v eno izmed predhodno definiranih kategorij.

Tekom procesa učenja smo preverili vpliv različnih oblik meta-podatkov na napovedno točnost modela, pri čimer je bilo naše ključno zanimanje usmerjeno v meta-podatke pridobljene s pomočjo sistema Never-Ending Language Learner, ki ga razvijajo na Univerzi Carnegie Mellon. Ugotovili smo, da slednji pozitivno vplivajo na uspešnost napovedovanja v kombinaciji z ostalimi meta-podatki. Za potrebe učenja smo uporabili algoritme logistična regresija, metoda podpornih vektorjev, naključni gozdovi in k-najbližjih sosedov. Izkazalo se je, da sta za gradnjo optimalnega modela najbolj primerna prva dva algoritma.

Obenem smo preizkusili tudi več pristopov k aktivnemu učenju, s katerimi lahko poenostavimo, pocenimo in pohitrimo proces ročnega označevanja novih primerov. Vsi preizkušeni pristopi so ponudili pozitiven rezultat, za najboljšega pa se je izkazal pristop, ki kombinira mero negotovosti napovedi

in koreliranosti med učnimi primeri.

Ključne besede: strojno učenje, odkrivanje znanj iz besedil, klasifikacija, Event Registry, Never-Ending Language Learner, aktivno učenje.

Abstract

Text mining, a field of machine learning that deals with the discovery of knowledge from text, is evolving rapidly. This fact has been recognized by the Artificial Intelligence Laboratory of Jožef Stefan Institute, which is developing a system called Event Registry that collects news articles from the Web in real-time, detects events therein and extracts relevant information. The component of the system which deals with the classification of articles into categories has not yet been fully developed. In a response to this, in our diploma thesis, we tried to upgrade a reference model. The results of our work have been positive, since we improved the predictive accuracy of classification of arbitrary news articles into one of the categories of our predefined taxonomy.

During the learning phase, we examined the impact of various forms of meta-data on the predictive accuracy of the model, where we focused mainly on meta-data obtained from Never-Ending Language Learner developed at Carnegie Mellon University. We assessed that the latter have a positive effect on the performance of the model if they are used in combination with other meta-data. For the purposes of learning we used different algorithms such as logistic regression, support vector machine, random forests and k-nearest neighbors. It turned out that the first two algorithms are the most appropriate for building the optimal predictive model.

At the same time, we also tested several approaches to active learning, by which we can simplify and speed up the process of manual labeling of new articles. All of them have produced a positive result, while approach

that combines uncertainty of prediction with correlation between learning instances proved to be the best.

Keywords: machine learning, text mining, classification, Event Registry, Never-Ending Language Learner, active learning.

Poglavje 1

Uvod

Količina informacij, ki jih človeštvo vsakodnevno proizvede, se povečuje z izjemno hitrostjo. To dejstvo nam prinaša očitne prednosti, kot so lažji dostop do znanja in pomoč pri hitrejšem sprejemanju odločitev, a pred nas hkrati postavlja velike, še do nedavnega, nepredstavljive izzive. Mednje spadajo zmnožnost hranjenja masovnih količih podatkov, sposobnost učinkovite ekstrakcije znanja iz zajetih vsebin ter smiselna, abstrahirana reprezentacija pridobljenega znanja.

Tu ključno vlogo igra *strojno učenje* (angl. *machine learning*), ki spada na področje *umetne inteligence* (angl. *artificial intelligence*). Ker so proizvedene informacije največkrat v tekstovni obliki, se za uspešno reševanje omenjenih izzivov poslužujemo metod ene od podzvrsti strojnega učenja, ki se ukvarja z *odkrivanjem znanj iz besedil* (angl. *text mining*).

Abstrakcije zajetih tekstovnih vsebin se lahko lotimo z *uvrščanjem* ali *klasifikacijo* (angl. *classification*) besedil v kategorije. To lahko storimo ročno, vendar je tako delo zamudno in ne prinaša velike dodane vrednosti. Nasproten pristop pa je avtomatski in temelji na metodah strojnega učenja, natančneje na procesu *nadzorovanega učenja* (angl. *supervised learning*). Točnost rezultata slednjega pristopa ni popolna, a vendar se splača žrtvovati del le-te za večjo učinkovitost in hitrost. Hkrati pa v prid tej trditvi govori tudi dejstvo, da je sposobnost avtomatske klasifikacije mogoče s časom obču-

tno izboljšati z metodo *aktivnega učenja* (angl. *active learning*), ki vključuje omejeno ročno participacijo človeškega faktorja pri vodenju učenja.

V Laboratoriju za umetno inteligenco na Institutu Jožef Stefan razvijajo sistem Event Registry, ki v realnem času zbira in analizira globalno objavljene spletne novice, le-te poskuša grupirati in v njih prepoznati dogodke ter iz njih avtomatsko ekstrahirati smiselne informacije. V okviru sistema razvijajo tudi algoritem za uvrščanje novičarskih člankov oz. novic v tekstovni obliki v predhodno definirane kategorije, ki zaenkrat še ni v aktivni uporabi. (V skladu z domeno našega problema so kategorije ekvivalentne tipu dogodkov). Tekom razvoja je bil izdelan prototip algoritma [8], ki smo ga s tem diplomskim delom skušali izboljšati in nadgraditi. Definirali smo enonivojsko taksonomijo, ki je obsegala osem različnih kategorij (npr. "umetnost in kultura", "oboroženi spopadi in napadi", "znanost in tehnologija"...). Na voljo smo imeli okoli 14.000 označenih (predhodno klasificiranih) novičarskih člankov v angleškem jeziku, nato pa smo naknadno pridobili še okoli 9000 prav tako označenih člankov. Na njihovi podlagi smo se naučili klasifikacije novih, še neoznačenih člankov (okoli 45.000), ki so bili prav tako v angleškem jeziku. Označeni članki so bili pridobljeni s portala dogodkov spletne enciklopedije Wikipedia ter preko uporabe spletnega programskega vmesnika britanske časopisne hiše The Guardian, neoznačeni članki pa zajeti s portala IJS Newsfeed, ki preko spletnega programskega vmesnika izpostavlja tok semantično obogatenih novic pridobljenih s pomočjo pregledovanja RSS spletnih strani.

Z drugimi besedami, zgradili smo napovedni model, ki predstavlja funkcijo oz. preslikavo $f : X \to Y$, ki zna novičarski članek X na vhodu preslikati oz. uvrstiti v eno od kategorij Y na izhodu. Pri tem smo se poslužili sledečih klasifikacijskih učnih algoritmov: *logistična regresija* (angl. *logistic regression*), *metoda podpornih vektorjev* (angl. *support vector machine*), *naključni gozdovi* (angl. *random forests*) in *metoda k-najbližjih sosedov* (angl. *k-nearest neighbors*). Članki so bili pred gradnjo modela predstavljeni v obliki *modela vreča besed* (angl. *bag-of-words model*), ki simbolizira klasičen

pristop pri *obdelavi naravnih jezikov* (angl. *natural language processing*) in *informacijskem poizvedovanju* (angl. *information retrieval*). Pri slednjem je besedilo predstavljeno kot neurejena zbirka besed, kjer je za vsak članek shranjena številnost določene besede. Model je bil nadalje obogaten s pomočjo spletnega programskega vmesnika sistema Enrycher z meta-podatki v obliki entitet zaznanih z algoritmom za *prepoznavanje imenskih entitet* (angl. *named entity recognition*) in ključnih besed pridobljenih s sistemom Event Registry, ki temeljijo na hierarhični taksonomiji za organiziranje spletnih strani v skupine ročno kuriranega portala DMoz ter s spletnim programskim vmesnikom časopisne hiše The Guardian. Na podlagi omenjenih entitet in ključnih besed smo nato s spletnega programskega vmesnika sistema NELL, ki neprestano bere splet in iz prebranih vsebin ekstrahira dejstva, pridobili še množico tipov entitet za posamezen članek. (Sistem NELL za tipe entitet uporablja termin kategorije, kar pa ima v tem diplomskem delu drug pomen). Pri gradnji modelov smo preizkušali različne kombinacije podatkov z meta-podatki in merili njihov učinek na klasifikacijo, predvsem pa nas je zanimal vpliv meta-podatkov pridobljenih s sistemom NELL na potencialno izboljšanje napovedne točnosti klasifikacijskega modela.

S postopkom gradnje optimalnega klasifikatorja za primer enonivojske taksonomije smo hkrati želeli izdelati vzorčni model, ki bi se ga dalo uporabiti pri specializaciji kategorij na različnih nivojih hierarhične taksonomije.

Na koncu smo z uporabo metod aktivnega učenja preizkusili še različne pristope izbire člankov, s katerimi obogatimo prvotni model in izboljšamo napovedno točnost. Uporabljena sta bila dva pristopa, ki temeljita na negotovosti napovedi, ter dva, kjer sta prvotna pristopa kombinirana s pristopom, ki meri podobnost med članki.

V nadaljevanju sledi opis strojnega učenja (poglavje 2), pri čimer posebno pozornost namenimo nadzorovanemu učenju in z njim povezanim pojmom klasifikacije, nato pa se poglobimo še v pristope k aktivnemu učenju in odkrivanje znanj iz besedil. V poglavju 3 sledi opis uporabljenih tehnologij in orodij, kjer si pogledamo sisteme Enrycher, IJS Newsfeed in Event Regi-

stry, portal dogodkov Wikipedia, spletni programski vmesnik The Guardian in sistem NELL. Obenem pozornost namenimo tudi uporabljenim programskim orodjem, ki so nam pomagala do rešitve zastavljene naloge. S poglavjem 4 pozornost usmerimo v opis poteka klasifikacije člankov iz novic v kategorije in s tem razlago lastne rešitve problema. V poglavju 5 predstavimo rezultate in podamo diskusijo le-teh. Diplomsko delo zaključimo s sklepnimi mislimi, kjer poudarek namenimo tudi potencialnim izboljšavam rešitve in možnostim za nadaljne delo (poglavje 6).

Poglavje 2

Strojno učenje

Strojno učenje je področje umetne inteligence, ki ga včasih povezujemo tudi s pojmom *odkrivanje znanj iz podatkov* (angl. *data mining*). Gre za pristop prepoznavanja vzorcev v podatkih s pomočjo algoritmov, ki se učijo in delajo napovedi nad podatki. Algoritmi tipično zgradijo napovedni model nad učnimi primeri za napovedovanje izbrane opazovane vrednosti na novih testnih primerih [6, 7].

Metode strojnega učenja lahko delimo glede na vrsto učenja [25]:

Nadzorovano učenje. Na podlagi označenih učnih primerov algoritem zgradi napovedni model, ki zna nove neoznačene testne primere preslikati v izhodno vrednost in s tem napovedati oznako. Oznake učnih primerov določi človek – "učitelj". Primera nadzorovanega učenja sta regresija in klasifikacija. Nadzorovano učenje podrobneje predstavitmo v poglavju 2.1.

Nenadzorovano učenje. Algoritmi *nenadzorovanega učenja* (angl. *unsupervised learning*) imajo na voljo le množico neoznačenih oz. nelabeliranih primerov, v katerih poskušajo prepoznati vzorce oz. strukturo. Na podlagi podobnosti vzorcev primere navadno združujejo v skupine. Značilnosti so avtomatsko izluščene iz podatkov, brez nadzora "učitelja". Primer takega učenja je *gručenje* (angl. *clustering*).

Vzpodbujevalno učenje. Pri *vzpodbujevalnem učenju* (angl. *reinforcement learning*) je algoritmu ponujena začetna množica učnih podatkov, na podlagi katere se uči preko sistema nagrajevanja in kaznovanja. Sistem temelji na dinamični interakciji z okoljem, v katerem mora agent izpolniti določen cilj, brez da bi mu "učitelj" povedal, če je ta cilj blizu, daleč ali že dosežen.

Druga delitev strojnega učenja pa se vrši na podlagi željene izhodne vrednosti sistema [1]:

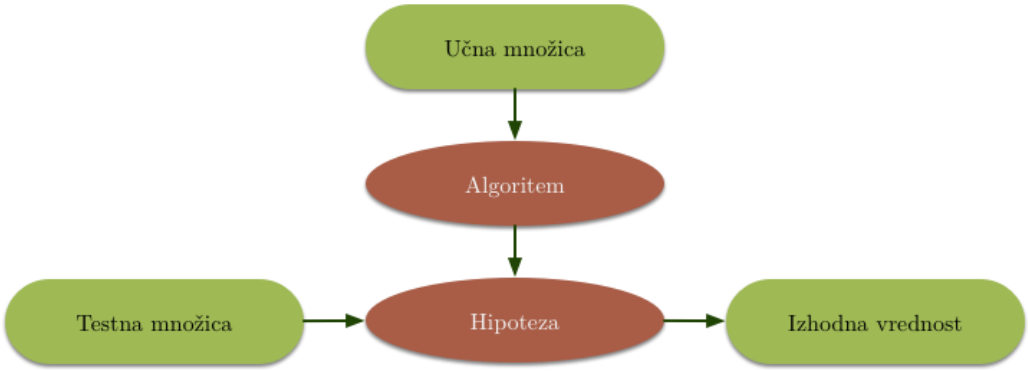
Klasifikacija. Tipično so vhodni učni primeri razdeljeni v dva ali več razredov, na njihovi podlagi pa se nato s pomočjo algoritma zgradi model, ki zna nove testne primere uvrstiti v enega ali več razredov. Izhodne vrednosti sistema so diskretne nominalne.

Regresija. Regresija je zelo podobna klasifikaciji, razlikuje se le v tem, da je željena izhodna vrednost sistema tu zvezna. Imenujemo jo odvisna regresijska spremenljivka, njena vrednost pa se spreminja z vrednostjo neodvisnih spremenljivk.

Gručenje. Množica učnih primerov je razdeljena v skupine šele med procesom učenja in ne pred tem, tako kot je to značilno za sisteme, ki delujejo po principu nadzorovanega učenja.

Ocena gostote verjetnosti. Izhodno vrednost sistema predstavlja distribucija vhodnih primerov v izbranem prostoru. Ocena gostote verjetnosti temelji na funkciji porazdelitvene gostote podatkov.

Znižanje dimenzionalnosti. Pristop omogoča poenostavitev vhodnih podatkov preko preslikave v nižjedimenzionalni prostor. Proces delimo v dva dela, in sicer izbiro *značilk* (angl. *features*) ter ekstrakcijo značilk.



Slika 2.1: Shema nadzorovanega učenja.

2.1 Nadzorovano učenje

Med procesom nadzorovanega učenja se želimo iz označenih (labeliranih) primerov v učni množici podatkov, s pomočjo učnega algoritma naučiti funkcije (preslikave, modela, hipoteze) [1, 6]. Učna množica $\mathcal{U}$ sestoji iz vhodnih primerov, ki so tipično predstavljeni kot vektorji, in njihovih željenih izhodnih vrednosti. Primeri testne množice $\mathcal{T}$ so predstavljeni v enaki obliki kot učni primeri, izhodno ciljno vrednost pa jim napovemo s pomočjo predhodno zgrajenega modela oz. hipoteze. Ta potek je shematsko predstavljen s sliko 2.1, kjer hipoteza oz. funkcija h slika primere iz vhodnega prostora X v izhodni prostor ocenjenih ciljnih vrednosti Y ($h : X \to Y$).

Učna in testna množica podatkov sta predstavljeni v vektorski (matrični) obliki:

$$X = \begin{bmatrix} x_1^{(1))} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{bmatrix}, \tag{2.1}$$

pri čimer vrstice matrike predstavljajo učne primere, stolpci pa značilke. Tipično z m označujemo število primerov, z n pa število značilk. $\vec{x^{(i)}}$ predstavlja i-ti učni primer v vektorski obliki, $x_j^{(i)}$ pa vrednost j-te značilke i-tega uč-

nega primera. Vsakemu učnemu primeru pripada tudi vektor izhodnih ciljnih vrednosti:

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix} \tag{2.2}$$

2.1.1 Klasifikacija

Klasifikacija je metoda, ki spada v skupino nadzorovanega učenja [1, 6, 7]. Pri njej skušamo primere vhodnega prostora preslikati v vrednosti izhodnega prostora s pomočjo hipoteze, ki je produkt izbranega učnega algoritma. Slednji preiskuje prostor vseh možnih hipotez in na podlagi izbrane *cenilne* ali *kriterijske funkcije* (angl. *cost function*) izbere najboljšo (tipično tisto, ki minimizira napako napovedi).

Pri klasifikaciji se lahko soočamo z dvorazrednim (binarnim) klasifikacijskim problemom, kjer $\vec{y} \in \{0, 1\}$ in velja, da je 0 negativni razred (-), ki označuje odsotnost ciljne vrednosti, 1 pa pozitivni razred (+), ki označuje prisotnost ciljne vrednosti.

Nismo omejeni le na dvorazredni problem, temveč se lahko soočimo tudi s problemom večrazredne narave. V slednjem primeru velja $\vec{y} \in \{0, 1, 2, 3...\}$. Zaradi mnoštva razredov ne moremo več govoriti o negativnem in pozitivnem razredu. Pri *večrazredni klasifikaciji* (angl. *multiclass classification*) zgradimo toliko klasifikatorjev, kot je število možnih ciljnih razredov. Gradnja posameznega klasifikatorja je v bistvu prirejena binarna klasifikacija, saj pri njej za primere pozitivnega razreda vzamemo le primere enega razreda, vse preostale primere pa tretiramo kot primere negativnega razreda. Ta pristop imenujemo tudi *eden proti vsem* (angl. *one-vs-all*). Pri napovedovanju razreda neoznačene testne primere preslikamo v razredno spremenljivko tako, da jim napovemo vrednost z vsakim posameznim klasifikatorjem, ki smo ga zgradili. Primeru dodelimo oznako tistega razreda, katerega klasifikator vrne največjo verjetnost.

Pri klasifikacijskih problemih lahko *neuravnoteženost* (angl. *skewness*) razredne spremenljivke povzroča težave [13, 14]. S tem se lahko spopademo tako, da utežimo primere različnih razredov z različnimi utežmi C. Slednji parameter nadzoruje vrednosti kazni za napačno klasificirane primere iz učne množice $\mathcal{U}$.

V enačbi

$$\sum_{j=0}^{n} C_j \sum_{\vec{x} \in \mathcal{U}_j} \psi(\vec{x}) \tag{2.3}$$

$\psi(\vec{x})$ predstavlja kazen za napačno klasifikacijo i-tega primera, C_j utež j-tega razreda, n število vseh razredov, $\mathcal{U}_j$ pa podmnožico $\mathcal{U}$, ki vsebuje primere, ki pripadajo razredu j. Uteži C določimo po formuli $\frac{m_-}{m_+} = \frac{C_+}{C_-}$, kjer m_- in m_+ predstavljata število vseh primerov v negativnem oz. pozitivnem razredu. Ta formula velja za dvorazredni klasifikacijski problem in jo izračunamo le enkrat. Pri večrazrednem klasifikacijskem problemu pa je potreben izračun za vsak razred posebej, t.j. j-krat. V primeru slednjega pri računanju uteži za razred j, primere, ki mu pripadajo, tretiramo kot pozitivni razred, primere vseh ostalih razredov pa kot negativni razred. Če za potrebe večrazredne klasifikacije uporabljamo pristop eden proti vsem, potem nam že sama intuicija pove, da je opisana sprememba v utežitvi napačno klasificiranih primerov različnih razredov pravilna.

Recimo, da imamo v $\mathcal{U}$ primere, ki pripadajo trem različnim razredom. V prvem razredu je 30 % primerov, v drugem 20 % in v tretjem 50 %. Utež prvega razreda tako zavzame vrednost $\frac{7}{3} \approx 2,33$, utež drugega $\frac{8}{2} = 4$, utež tretjega pa $\frac{5}{5} = 1$. Vrednosti uteži razreda j so torej obratno sorazmerne s številom primerov, ki pripadajo razredu j pri gradnji j-tega klasifikatorja.

Na neuravnoteženost razredne spremenljivke moramo paziti tudi ko delimo začetno množico primerov na učno, validacijsko in testno ter pri postopku *internega prečnega preverjanja* (angl. *cross-validation*) (glej poglavje 2.1.2). Pomembno je, da je porazdelitev (distribucija) razredov v vseh množicah, ki jih pridelamo, približno enaka. Razlog za to tiči v dejstvu, da mora biti testna množica, na kateri testiramo model zgrajen na učni množici, re-

prezentativen vzorec učne množice, ki predstavlja populacijo. Postopek, s katerim to dosežemo, se imenuje *stratificirano vzorčenje* (angl. *stratified sampling*) [3]. Zanj velja, da sprva razdelimo primere istih razredov v posamezne homogene podmnožice, šele nato pa naključno vzorčimo primere znotraj vsake podmnožice.

2.1.1.1 Logistična regresija

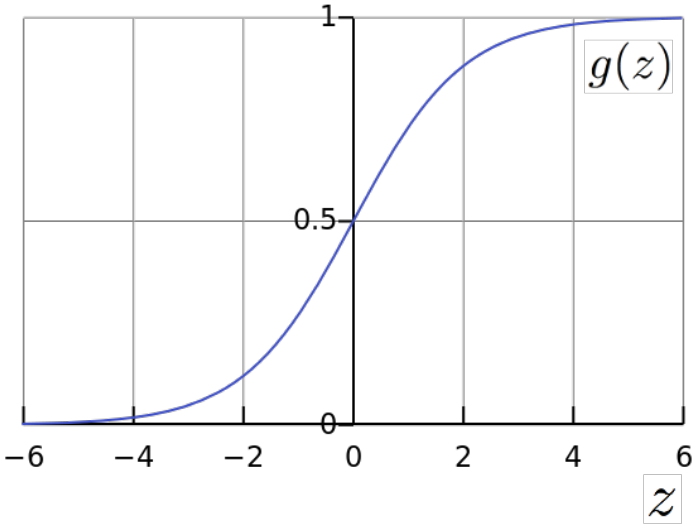
Logistična regresija je eden najpreprostejših klasifikacijskih algoritmov [1, 25, 7]. Uporablja se za potrebe nadzorovanega učenja. Z uporabo algoritma skušamo najti optimalno *ločitveno mejo* (angl. *decision boundary*) med primeri, ki pripadajo različnim razredom. Slednja torej ločuje *vektorski prostor* (angl. *vector space*) učne množice na toliko vektorskih podprostorov oz. podmnožic, kolikor je različnih razredov. Ločitvena meja obenem predstavlja območje oz. vektorski podprostor, kjer je izhodna vrednost klasifikatorja dvoumne narave. Če je ločitvena meja *hiperravnina* (angl. *hypersurface)*, potem je klasifikacijski problem linearen, razredi pa so linearno ločljivi. V nasprotnem primeru govorimo o nelinearnem klasifikacijskem problemu, kjer je ločitvena meja neka druga oblika vektorskega podprostora, razredi pa niso linearno ločljivi. Primeri take ločitvene meje so hipoteze, ki vsebujejo značilke polinomov višjih razredov.

Definirajmo vektor parametrov hipoteze

$$\vec{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}, \tag{2.4}$$

čigar vrednosti morajo biti izbrane tako, da slednja čim bolje loči primere, ki pripadajo različnim razredom. Enačba

$$z = \theta_0 \; x_0 + \theta_1 \; x_1 + \ldots + \theta_n \; x_n = \sum_{j=0}^{n} \theta_j \; x_j = \vec{\theta}^{\,T} \vec{x} \tag{2.5}$$



Slika 2.2: Sigmoidna (logistična) funkcija $g(z)$.

predstavlja skalarni produkt med vektorjem parametrov $\vec{\theta}$ in vektorjem značilk posameznega učnega primera $\vec{x}$. Če zgornjo enačbo zavijemo v sigmoidno (logistično) funkcijo (slika 2.2)

$$g(z) = \frac{1}{1 + e^{-z}}, \tag{2.6}$$

dobimo enačbo hipoteze algoritma

$$h_{\vec{\theta}}(\vec{x}) = g(\vec{\theta}^{\,T}\vec{x}) = \frac{1}{1 + e^{-\vec{\theta}^{\,T}\vec{x}}} \tag{2.7}$$

Izhodna vrednost hipoteze

$$h_{\vec{\theta}}(\vec{x}) = P(y = 1 \mid \vec{x};\ \vec{\theta}) \tag{2.8}$$

je zvezno realno število med 0 in 1. Njeno vrednost lahko interpretiramo kot verjetnost P, da je preslikava primera $\vec{x}$ ob danih parametrih $\vec{\theta}$ zavzela vrednost 1, kar je ekvivalentno pozitivnemu razredu.

Če nek primer iz učne množice spada v pozitivni razred, želimo, da bi bila vrednost hipoteze $h_{\vec{\theta}}(\vec{x})$ enaka 1, to pa je takrat, ko velja $\vec{\theta}^{\,T}\vec{x} \gg 0$. In obratno, če nek primer spada v negativni razred, želimo, da bi bila vrednost

hipoteze $h_{\vec{\theta}}(\vec{x})$ enaka 0, to pa je takrat, ko velja $\vec{\theta}^{\,T}\vec{x} \ll 0$. Mejni primeri okoli vrednosti $\vec{\theta}^{\,T}\vec{x} = 0$ ležijo v podprostoru ločitvene meje in jih je zato težje pravilno klasificirati.

Za boljše razumevanje lahko navedemo primer, ko želimo s klasifikacijo, na podlagi različnih merjenih značilk (npr. velikost tumorja, starost osebe ipd.) napovedati, ali ima oseba benigni (negativni razred oz. 0) ali maligni (pozitivni razred oz. 1) tumor. Če npr. hipoteza $h_{\vec{\theta}}(\vec{x})$ pri danih parametrih $\vec{\theta}$ zavzame vrednost 0.7, to lahko interpretiramo tako, da je verjetnost, da je opazovani tumor maligni enaka 70 %.

Optimizacija vektorja parametrov $\vec{\theta}$ oz. preiskovanje hipoteznega prostora se lahko izvaja z metodo *gradientnega spusta* (angl. *gradient descent*), ki v najboljšem primeru konvergira v globalni minimum, sicer pa v lokalnega [11]. Metoda skuša minimizirati kriterijsko funkcijo $J(\vec{\theta})$:

$$min\ J(\vec{\theta}) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)}\ log(h_{\vec{\theta}}\ (\vec{x^{(i)}}))\ +\ (1-y^{(i)})\ log(1-h_{\vec{\theta}}\ (\vec{x^{(i)}}))] \quad (2.9)$$

Izbira optimalnih vrednosti vektorja parametrov $\vec{\theta}$ predstavlja glavni cilj algoritma.

Za preprečevanje prevelikega *prilagajanja* hipoteze *učnim podatkom* (angl. *overfitting*) in posledično slabši generalizaciji na testnih podatkih, se izvaja postopek *regularizacije* (angl. *regularization*). Kriterijski funkciji $J(\vec{\theta})$ prištejemo dodatni člen

$$\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2, \quad (2.10)$$

ki povzroči, da ostanejo vrednosti vektorja parametrov $\vec{\theta}$ nizke. Za primerno izbiro parametra λ, ki določa stopnjo regularizacije, se tipično uporablja interno prečno preverjanje (glej poglavje 2.1.2).

2.1.1.2 Metoda podpornih vektorjev

Metoda podpornih vektorjev oz. SVM spada v skupino algoritmov, ki se uporabljajo v okviru nadzorovanega učenja [1, 6, 7, 11]. V svoji osnovi je zelo

podoben logistični regresiji, vendar se je sposoben naučiti veliko bolj kompleksnih nelinearnih funkcij. Zaradi slednjega je zelo razširjen v sistemih strojnega učenja, še posebej na področju obdelave naravnih jezikov, kjer imajo učne množice zaradi narave problemov tipično zelo veliko značilk. Algoritem SVM pri učenju uporabi vse, tudi manj pomembne, značilke. Pomemben je predvsem način kombiniranja značilk in ne njihova izbira. Tipično metode SVM dosegajo visoke točnosti napovedi, na drugi strani pa je njihova slabost otežena interpretacija naučenega modela.

SVM lahko z drugimi besedami opišemo tudi kot *klasifikator, ki deluje po principu največjega roba* (angl. *large margin classifier*). Glavna ideja je ločevanje primerov različnih razredov z ločitveno mejo, ki je čim večja/širša, kar pomeni, da mora biti razdalja med ločitveno mejo in najbližjim pozitivnim primerom na eni strani in najbližjim negativnim primerom na drugi strani maksimalna. Nagnjenost k doseganju čim večje razdalje med učnimi primeri različnih razredov, daje algoritmu določeno mero robustnosti in boljšo generalizacijo pri klasifikaciji testnih primerov.

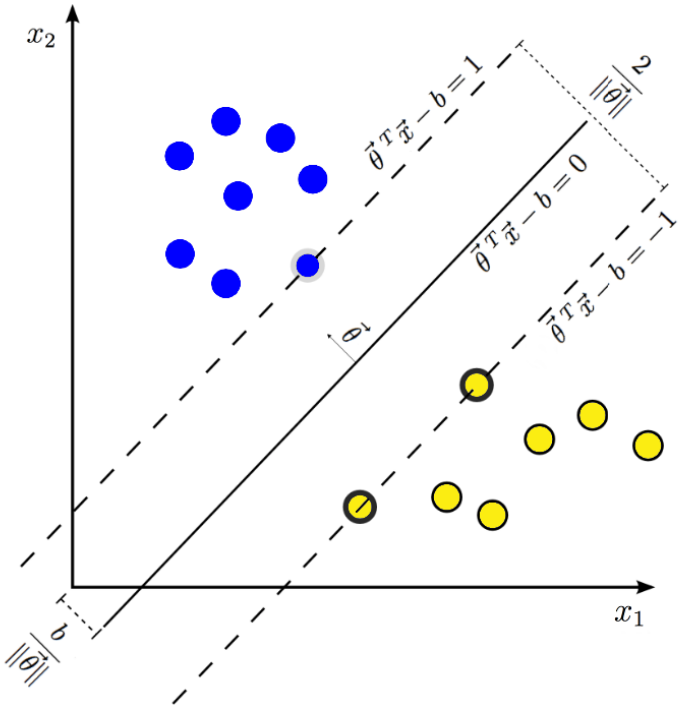
Za linearno ločljive vektorske prostore (učne množice), lahko pri gradnji modela uporabimo t.i. *linearno jedro* (angl. *linear kernel*). Definirajmo učno množico $\mathcal{U}$

$$\mathcal{U} = \{(\vec{x^{(i)}},\ y^{(i)}) \mid \vec{x^{(i)}} \in \mathbb{R},\ y^{(i)} \in \{-1, 1\}\};\ i \in \{1, 2, ...m\}, \tag{2.11}$$

ki vsebuje m označenih primerov $\vec{x^{(i)}}$ v p-dimenzionalni vektorski obliki, katerim pripadajo oznake $y^{(i)}$, ki so bodisi -1 bodisi 1. Naloga algoritma je poiskati hipotezo (v obliki hiperravnine)

$$h_{\vec{\theta}}(\vec{x}) = \vec{\theta}^{\,T}\vec{x} - b = 0, \tag{2.12}$$

kjer $\vec{\theta}$ predstavlja vektor parametrov modela in hkrati normalo hiperravnine, *prag* (angl. *bias*) b pa razdaljo hiperravnine od koordinatnega izhodišča v smeri normale. Torej, če je $\mathcal{U}$ linearno ločljiva, lahko izberemo dve hiperravnini tako, da popolnoma ločita podatke različnih razredov in da je razdalja med njima maksimalna. Vmesno področje imenujemo rob, katerega širina predstavlja zanesljivost klasifikacije primera iz testne množice $\mathcal{T}$.



Slika 2.3: Ločitvena meja med razredoma, omejena s hiperravninama, določena z algoritmom SVM.

Ideja principa največjega roba temelji na podpornih vektorjih, od koder tudi ime algoritma. To so tisti učni primeri, ki ležijo na eni oz. blizu ene izmed hiperravnin in imajo na lego ločitvene meje največji vpliv. Daljši kot so podporni vektorji, širša je ločitvena meja oz. razdalja med obema hiperravninama. Na sliki 2.3 je prikazan primer ločitvene meje med dvema razredoma, ki jo omejujeta izbrani hiperravnini. Pri tem so učni primeri $x^{\vec{(i)}}$ opisani zgolj z dvema značilkama (x_1 in x_2). Posebna podmnožica primerov so podporni vektorji, ki ležijo na robu ene izmed obeh hiperravnin in so na sliki 2.3 posebej označeni.

Spomnimo se, da za primer iz učne množice, ki spada v pozitivni razred, želimo vrednost napovedi $h_{\vec{\theta}}(\vec{x})$, ki bi bila enaka 1, za kar mora veljati $\vec{\theta}^{\,T}\vec{x} \gg 0$. In obratno za primere, ki spadajo v negativni razred, želimo vrednost

napovedi $h_{\vec{\theta}}(\vec{x})$, ki bi bila enaka 0, kar pa velja, ko je $\vec{\theta}^{\,T}\vec{x} \ll 0$. Pri SVM je vpeljan še dodatni varnostni faktor, t.j. *robni faktor* (angl. *margin factor*). Zanj velja, da prestavi mejo za klasifikacijo v pozitivni oz. negativni razred, saj mora v prvem primeru veljati $\vec{\theta}^{\,T}\vec{x} - b \geq 1$, v drugem pa $\vec{\theta}^{\,T}\vec{x} - b \leq -1$. S tem preprečimo, da bi primeri padli v območje med obema hiperravninama. Optimalna hiperravnina mora zadovoljiti sledeče omejitve:

$$y^{(i)}\ (\vec{\theta}^{\,T}\vec{x^{(i)}} - b) \geq 1;\ \ i \in \{1, 2, ...m\} \tag{2.13}$$

Geometrično gledano je razdalja med obema hiperravninama enaka $\frac{2}{\|\vec{\theta}\|}$, kjer je $\|\vec{\theta}\|$ dolžina normale hiperravnine. Za maksimizacijo te razdalje moramo torej minimizirati $\|\vec{\theta}\|$. Tako definiramo optimizacijski problem algoritma:

$$\underset{(\vec{\theta},\ b)}{arg\ min}\ \frac{1}{2}\ \|\vec{\theta}\|^2 \tag{2.14}$$

S tem, ko minimiziramo velikost koeficientov hiperravnine $\vec{\theta}$, zmanjšujemo kompleksnost rešitve. Problem minimizacije z omejitvami (enačba (2.13)) se prevede na problem maksimizacije funkcionala

$$\underset{(\vec{\theta},\ b)}{arg\ min}\ \underset{\alpha \geq 0}{max}\ \{\frac{1}{2}\ \|\vec{\theta}\|^2 - \sum_{i=1}^{m} \alpha_i\ [y^{(i)}\ (\vec{\theta}^{\,T}\vec{x^{(i)}} - b) - 1]\} \tag{2.15}$$

Slednji se rešuje s pomočjo *kvadratičnega programiranja* (angl. *quadratic programming*), dočim podrobnejšo razlago najdemo v [6, 7, 11].

V primeru da se vektorskega prostora ne da linearno ločiti, preslikamo učne primere v implicitni višjedimenzionalni (lahko tudi neskončnodimenzionalni) prostor s pomočjo ene od *jedrnih metod* (angl. *kernel methods*). Tako postanejo primeri različnih razredov linearno ločljivi. Za to transformacijo poskrbijo jedra oz. posebne funkcije, ki računajo podobnost med pari primerov v višjedimenzionalnem prostoru. Primeri jedrnih funkcij so Gaussova, polinomska, χ^2, radialna, sigmoidna itd. Od izbire primernega jedra je odvisna uspešnost metode, dočim se v tej nalogi omejimo na linearna jedra.

2.1.1.3 Naključni gozdovi

Naključni gozdovi so algoritem strojnega učenja, ki po principu večkratne razlage zgradi mnogo različnih hipotez [6, 7]. Slednje na različne načine modelirajo podatke. Algoritem je namenjen izboljšanju napovedne točnosti algoritma *odločitvenih dreves* (angl. *decision trees*). Temelji na metodah angl. *bagging*, krajše za angl. *bootstrap aggregating*, in *glasovanju* (angl. *voting*). *Bagging* skrbi za to, da isti učni algoritem poženemo večkrat z različno učno množico $\mathcal{U}$, glasovanje pa služi kombiniranju napovedi različnih hipotez v končno napoved.

Bagging je metoda razmnoževanja učnih primerov, s katero generiramo različne $\mathcal{U}$. Dobro deluje predvsem pri algoritmih z visoko varianco, med katere spadajo odločitvena drevesa. Na podlagi slednje ugotovitve so razvili algoritem naključni gozdovi. Varianca je v tem primeru napaka, ki izvira iz učnih podatkov in višja kot je, bolj občutljiv je algoritem na podatke v $\mathcal{U}$. Visoka varianca je posledica velike globine odločitvenih dreves. Taki modeli so močno nagnjeni k prevelikemu prilagajanju učnim podatkom (tudi *osamelcem* (angl. outliers), ki predstavljajo šum v podatkih). Gradnja več odločitvenih dreves namesto enega zmanjšuje varianco končnega kombiniranega modela, brez da bi se povečala njegova *pristranskost* (angl. *bias*). Četudi so napovedi posameznega odločitvenega drevesa občutljive na šum v učnih podatkih, to ne velja za povprečje napovedi več odločitvenih dreves, pod pogojem, da modeli med seboj niso korelirani. Ta pogoj je pri algoritmu naključni gozdovi izpolnjen, saj za razliko od odločitvenih dreves ne deluje na deterministični, temveč verjetnostni (probabilistični) osnovi.

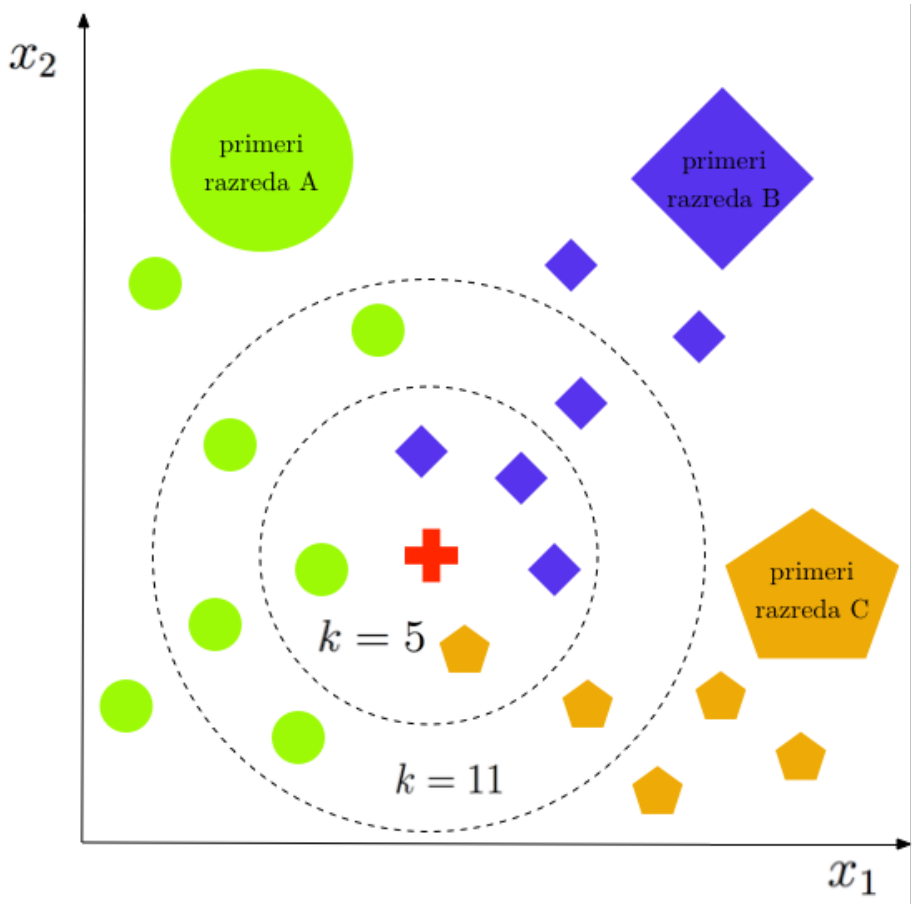
Pri naključnih gozdovih se metode *bagging* poslužimo tako, da v vsaki iteraciji zgradimo eno odločitveno drevo na podlagi različne $\mathcal{U}$. Postopek gradnje i-tega odločitvenega drevesa je sledeč: v začetni učni množici $\mathcal{U}$ imamo m primerov, zato v i-to učno množico $\mathcal{U}_i$ m-krat naključno izberemo primer z vračanjem, zaradi česar se lahko isti učni primer v $\mathcal{U}_i$ večkrat ponovi, določeni primeri iz $\mathcal{U}$ pa se ne bodo znašli v nobeni od zgrajenih $\mathcal{U}_i$. Na podlagi $\mathcal{U}_i$ se naučimo i-tega modela oz. hipoteze. Označimo množico vseh

atributov z $\mathcal{A}$, njeno moč pa z $n_{\mathcal{A}}$. Pri gradnji i-tega odločitvenega drevesa je vpeljana dodatna mera naključnosti, saj pri izbiri najboljšega atributa v vsakem vozlišču navadno ne izberemo zgolj enega, tj. najbolj informativnega, temveč naključno podmnožico atributov $\mathcal{B} \subset \mathcal{A}$, ki vstopajo v izbor za najboljši atribut. Pri klasifikacijskih problemih se v $\mathcal{B}$ tipično naključno izbere $n_{\mathcal{B}} = \log n_{\mathcal{A}} + 1$ atributov. Če velja $n_{\mathcal{B}} = 1$, potem so vse izbire atributov v vozliščih odločitvenih dreves popolnoma naključne. Temu postopku z drugimi besedami rečemo tudi *atributni bagging* (angl. *feature bagging*). Tekom učenja končnega napovednega modela izvedemo toliko iteracij, kolikor odločitvenih dreves si želimo. Označimo to število s q. Vsako od odločitvenih dreves se uporabi za klasifikacijo novih testnih primerov po metodi glasovanja. Pri tem ima vsako drevo po en glas, ki ga nameni razredu, kamor bi samo klasificiralo primer. Primer je uvrščen v razred z največ glasovi.

Tipično se q giblje okoli 100 ali več, odvisno od velikosti začetne $\mathcal{U}$ in narave problema. V praksi se izkaže, da večji q daje boljšo napovedno točnost, saj se s povprečenjem zmanjšuje preveliko prilagajanje modela $\mathcal{U}$. A vendar q ni moč povečevati v neskončnost, saj pri določeni vrednosti limitira – napovedna točnost se z zviševanjem vrednosti q ne spreminja več. Kljub temu naključni gozdovi spadajo med algoritme strojnega učenja, ki dosegajo najvišje napovedne točnosti. Slabost metode pa je v težki interpretaciji odločitev napovednega modela, kar je posledica velike, nepregledne, naključno zgrajene množice odločitvenih dreves.

2.1.1.4 Metoda k-najbližjih sosedov

Metoda k-najbližjih sosedov oz. k-NN spada med najpreprostejše algoritme strojnega učenja [6, 7, 11]. Uporablja se tako za klasifikacijske, kot tudi regresijske probleme. Metodo zaznamuje koncept imenovan *leno učenje* (angl. *lazy learning*), kar pomeni, da se modela sploh ne naučimo oz. da faza učenja sploh ne obstaja. Posledično moramo imeti pri napovedovanju venomer na razpolago učno množico primerov $\mathcal{U}$, saj le-te ne abstrahiramo z modelom. Leno učenje nam po eni strani prihrani čas učenja, po drugi pa prinaša ve-



Slika 2.4: Uvrščanje primera označenega z rdečim križcem v enega od treh razredov z algoritmom k-NN.

liko časovno zahtevnost pri klasifikaciji novih primerov. Pri napovedovanju iščemo podmnožico podobnih primerov – najbližjih sosedov $\mathcal{Z}$. Moč $\mathcal{Z}$ določa faktor k, za katerega ne moremo reči, da je parameter modela, temveč parameter napovedovanja. Izbrano $\mathcal{Z}$ najbolj podobnih primerov uporabimo za napoved razreda. Pri klasifikacijskih problemih je za napoved razreda primerov v testni množici $\mathcal{T}$ uporabljeno glasovanje, pri čimer je napovedan razred, ki dobi največ glasov, pri regresijskih problemih pa je napovedana povprečna vrednost regresijske spremenljivke vseh primerov v $\mathcal{Z}$.

Pri najpreprostejši različici algoritma k-NN za napovedovanje razreda

$y^{(i)}$ primeru $x^{(i)}$ iz $\mathcal{T}$ v celotni $\mathcal{U}$ poiščemo k najbolj podobnih oz. najbližjih primerov (glej spodaj), katere uvrstimo v $\mathcal{Z}$. Na podlagi primerov v $\mathcal{Z}$ primeru $x^{(i)}$ napovemo večinski razred:

$$y^{(i)} = \arg\max_{y \in \mathcal{Y}} \sum_{z \in \mathcal{Z}} \delta(y,\ y_z), \tag{2.16}$$

pri čimer je z $\mathcal{Y}$ označena množica vseh razredov, δ pa predstavlja Kroneckerjev delta. Slika 2.4 prikazuje primer klasifiakcije primera, kjer velja $Y = 3$ in $k \in \{5, 11\}$. Ko je k enak 5 je primeru pripisan razred B, ko pa je k enak 11, primer uvrstimo v razred A.

Za parameter k pri klasifikaciji tipično izberemo neko manjše liho število, saj se tako izognemo situacijam, kjer bi bil izid glasovanja za določen primer neodločen. Če podatki ne vsebujejo šuma, je za k najbolje izbrati 1. S povečevanjem deleža primerov, ki predstavljajo napako v podatkih, pa je smiselno premo sorazmerno povečevati tudi parameter k. Večji kot je k, več vrednosti povprečimo in s tem zmanjšamo verjetnost, da so vsi primeri v $\mathcal{Z}$ napaka v podatkih. A vendar večji k povečuje možnost, da se v $\mathcal{Z}$ znajdejo primeri, ki sploh niso podobni izbranemu primeru $x^{(i)}$.

Med metrike, ki se uporabljajo za računanje bližine med primeri spadajo Manhattanska razdalja:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|, \tag{2.17}$$

evklidska razdalja [11]:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.18}$$

in razdalja Minkowskega:

$$d(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p}, \tag{2.19}$$

ki predstavlja generalizacijo prvih dveh. Če p zavzame vrednost 1, je razdalja Minkowskega enaka Manhattanski razdalji, v primeru, da je p enak 2, pa evklidski razdalji.

Ena od nadgradenj osnovne metode je algoritem, ki ga imenujemo *razdaljno uteženih k-najbližjih sosedov* (angl. *distance-weighted k-nearest neighbors*) [6, 7]. Ta različica je bolj robustna, zanjo pa je značilno, da utežimo vpliv učnih primerov na napoved obratno sorazmerno z razdaljo. Razdalja lahko na napoved vpliva linearno, kvadratno, eksponentno itd.

V primeru enačbe

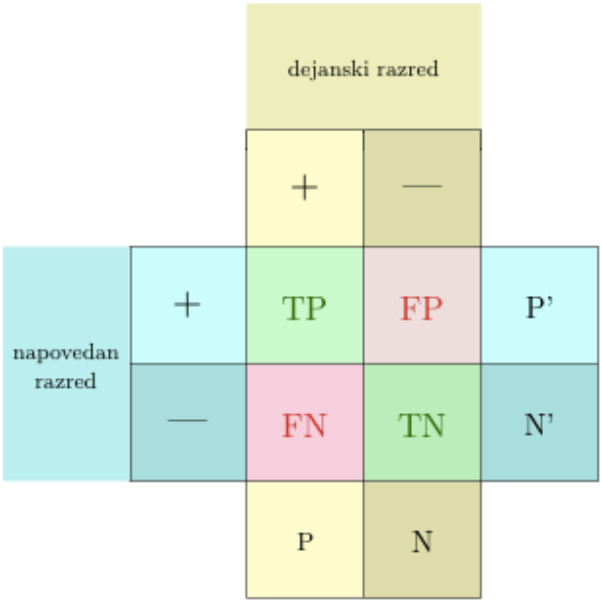
$$y^{(i)} = \underset{y \in \mathcal{Y}}{\arg\max} \sum_{z \in \mathcal{Z}} \frac{\delta(y,\ y_z)}{d(x^{(i)}, z)^2} \tag{2.20}$$

je izbran kvadratni vpliv, pri čimer $d(x^{(i)}, z)^2$ predstavlja s kvadratom obteženo razdaljo med primerom $x^{(i)}$, za katerega napovedujemo razred, in primerom z iz podmnožice najbližjih sosedov $\mathcal{Z}$. Tipično je k enak številu učnih primerov, kar pomeni, da ga ne omejujemo in da na napoved vplivajo vsi primeri. To si lahko privoščimo, saj je vpliv oddaljenih primerov na končno napoved zanemarljiv.

2.1.2 Validacija in merjenje uspešnosti klasifikacije

Množico vseh primerov lahko delimo na učno $\mathcal{U}$, validacijsko $\mathcal{V}$ in testno $\mathcal{T}$ v nekem fiksno določenem razmerju. $\mathcal{V}$ se uporablja za določitev optimalnega nabora parametrov modela zgrajenega na $\mathcal{U}$, $\mathcal{T}$ pa le za končno testiranje najboljšega modela. Ta končna ocena nam podaja tudi predvideno uspešnost napovedovanja na še ne videnih podatkih. Tipično se začetna množica deli v razmerju $\mathcal{U} : \mathcal{V} : \mathcal{T} = 60 : 20 : 20$. Primere v $\mathcal{U}$ lahko pred gradnjo napovednega modela še naključno premešamo.

Pogosto se za ocenjevanje optimalne kombinacije parametrov modela uporablja tudi interno prečno preverjanje [7]. Pri tem originalno množico označenih primerov sprva razdelimo na $\mathcal{U}$ in $\mathcal{T}$ v nekem fiksno določenem razmerju, npr. $\mathcal{U} : \mathcal{T} = 70 : 30$. $\mathcal{U}$ se nato uporabi za postopek internega prečnega preverjanja, pri čimer se razdeli na k delov. Opravi se k iteracij, v vsaki posamezni pa se za i-to testno množico $\mathcal{T}_i$ vzame enega od k delov, preostalih $k-1$ delov pa predstavlja i-to učno množico $\mathcal{U}_i$, na kateri se zgradi i-ti napovedni

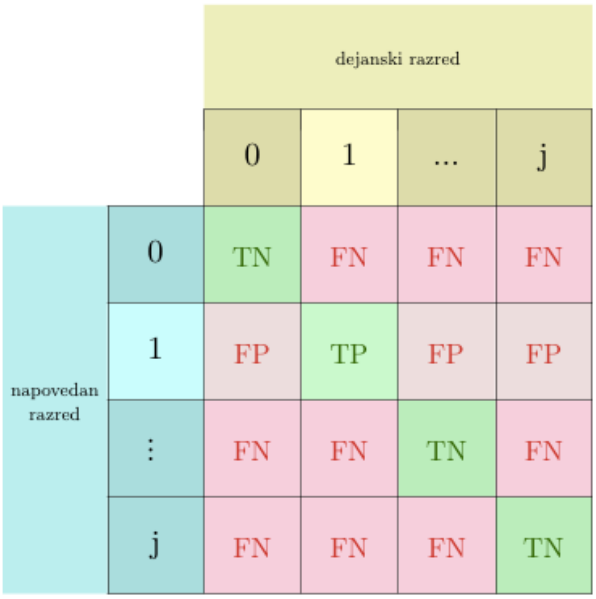


Slika 2.5: Kontingenčna matrika za dvorazredni klasifikacijski problem.

model. S slednjim napovemo razrede za primere v $\mathcal{T}_i$, ki jih sproti shranjujemo v enoten vektor (k-krat dopolnjen). Na podlagi vektorja napovedanih razredov z izbranimi metrikami nato merimo uspešnost klasifikacije.

Uspešnost klasifikacije lahko preverimo s *kontingenčno matriko* (angl. *confusion matrix*), ki tabelarično prikazuje, koliko primerov je bilo pravilno oz. napačno klasificiranih glede na resničen razred [7]. Na sliki 2.5 je prikazana kontingenčna matrika za dvorazredni klasifikacijski problem. Vrednosti na diagonali – *pravilno klasificirani pozitivni primeri* (angl. *true positives*) oz. TP in *pravilno klasificirani negativni primeri* (angl. *true negatives*) oz. TN – predstavljajo pravilne napovedi klasifikatorja, preostale – *napačno klasificirani pozitivni primeri* (angl. *false positives*) oz. FP in *napačno klasificirani negativni primeri* (angl. *false negatives*) oz. FN – pa napačne. S P in N so označeni seštevki resnično pozitivnih oz. negativnih primerov, medtem ko P' in N' predstavljata seštevka napovedanih pozitivnih oz. negativnih primerov.

Podobno je v primeru večrazrednega klasifikacijskega problema, kjer nimamo več dveh razredov, temveč j. Slika 2.6 prikazuje kontingenčno matriko



Slika 2.6: Kontingenčna matrika za večrazredni klasifikacijski problem.

večrazredne klasifikacije. Velja, da so na diagonali pravilne napovedi klasifikatorja, ostale pa so napačne.

Klasifikacijska točnost (angl. *classification accuracy*) oz. CA je mera, ki predstavlja razmerje med številom pravilno klasificiranih primerov in številom vseh primerov [1, 6, 7]:

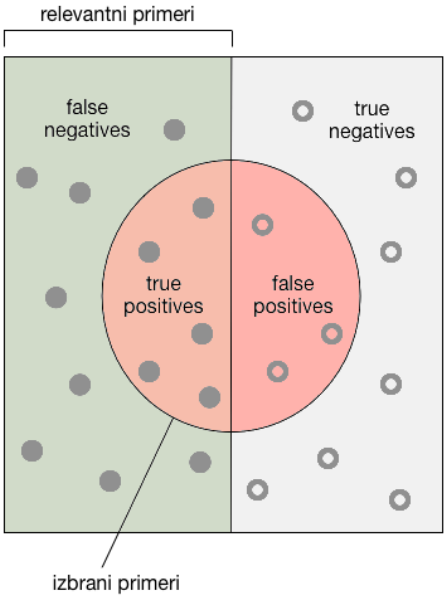
$$CA = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.21}$$

Mera je občutljiva na distribucijo razredov, zato je lahko pri večrazrednih klasifikacijskih problemih manj informativna.

Natančnost (angl. *precision*) je mera, ki nam podaja razmerje med pravilno klasificiranimi pozitivnimi primeri in vsemi pozitivno klasificiranimi primeri [1, 6, 7]:

$$natančnost = \frac{TP}{TP + FP} \tag{2.22}$$

Mero se da najbolje ponazoriti na primeru informacijskega poizvedovanja



Slika 2.7: Natančnost in priklic v informacijskem poizvedovanju.

(slika 2.7). Natančnost predstavlja delež izbranih primerov (tisti, ki jih je klasifikator označil za pozitivne), ki so relevantni (tisti, ki so v resnici pozitivni).

Priklic ali *občutljivost* (angl. *recall*) je mera, ki nam podaja razmerje med pravilno klasificiranimi pozitivnimi primeri in vsemi v resnici pozitivnimi primeri [1, 6, 7]:

$$priklic = \frac{TP}{TP + FN} \tag{2.23}$$

Mero se ponovno da najbolje ponazoriti na primeru informacijskega poizvedovanja (slika 2.7). Priklic predstavlja delež relevantnih primerov (tisti, ki so v resnici pozitivni), ki so bili izbrani (tisti, ki jih je klasifikator označil za pozitivne).

F-mera (angl. *F-measure* ali *F-score*) upošteva hkrati natančnost in priklic in predstavlja njuno uteženo povprečje. Pogosto se uporablja na področju informacijskega poizvedovanja za merjenje učinkovitosti spletnih iskalnih

sistemov in klasifikacije dokumentov [1, 6, 7]. Definirana je kot

$$F_\beta\text{-}mera = (1 + \beta^2) * \frac{natančnost * priklic}{(\beta^2 * natančnost) + priklic}, \tag{2.24}$$

pri čimer parameter β določa težo natančnosti oz. priklica. Najbolj pogosto se za parameter β uporablja vrednost 1:

$$F_1\text{-}mera = 2 * \frac{natančnost * priklic}{natančnost + priklic}, \tag{2.25}$$

kjer imata natančnost in priklic enako težo. F1-mera v tem primeru predstavlja harmonično povprečje natančnosti in priklica. Definicijsko območje F1-mere je $[0, 1]$. Pogosto za β izberemo tudi vrednosti 0,5 in 2. Pri prvi velja, da je večji pomen oz. teža dana natančnosti, pri drugi pa priklicu.

Natančnost, priklic in F-mera pri izračunu ne upoštevajo potencialno velikega števila TN, saj nas tipično na področju informacijskega poizvedovanja le-ti ne zanimajo.

2.2 Aktivno učenje

Aktivno učenje predstavlja poseben primer delno nadzorovanega učenja oz. njegovo generalizacijo [4, 29]. Pri njem je učni algoritem v interakciji s človekom ali pa nekim drugim dodatnim virom informacij s ciljem pridobitve željenih ciljnih vrednosti (v primeru klasifikacije so to razredi) za neoznačene primere. Tipično se aktivno učenje uporablja pri problemih strojnega učenja, kjer je na voljo obilica neoznačenih primerov in le majhen delček označenih. Taki primeri so npr. odkrivanje goljufij v finančnem svetu, klasifikacija spletnih strani in napovedovanje strukture proteinov. Ker večja količina označenih primerov v splošnem daje boljše napovedne točnosti zgrajenih modelov, si v takih primerih želimo pridobiti nove označene primere. Ker je ročno označevanje zelo počasna in hkrati draga operacija, morajo biti primeri, ki so ponujeni človeškemu ocenjevalcu smotrno izbrani. Z drugimi besedami, želimo si novih primerov, ki bodo kar se da dobro vplivali na napovedno točnost modela. Za izbiro trenutnemu modelu najbolj informativnih

primerov obstajata dve skupini pristopov. Prva skupina temelji na izbiranju primerov na podlagi *negotovosti* (angl. *uncertainty*) napovedi klasifikatorja, kjer se upoštevajo verjetnosti, s katerimi klasifikator uvršča dan primer v posamezen razred. Druga skupina kombinira mere negotovosti z mero, ki temelji na vsoti podobnosti posameznega primera z vsemi ostalimi primeri v množici neoznačenih primerov.

Na voljo imamo torej označeno množico primerov $\mathcal{U}^O$ ter neoznačeno množico primerov $\mathcal{U}^N$. V $\mathcal{U}^N$ želimo izbrati podmnožico, ki bo z obogatitvijo $\mathcal{U}^O$ maksimizirala napovedno točnost modela. Označimo metriko, ki izbira primere iz $\mathcal{U}^N$ na podlagi negotovosti napovedi s f, metriko, ki upošteva *korelacije* (angl. *correlations*) oz. podobnosti med primeri pa z g. Kombinirana metrika, ki podaja *koristnost* (angl. *utility*) posameznega primera naj nosi oznako u.

Prvi primer metrike negotovosti napovedi f je *rob* (angl. *margin*):

$$f_{rob} = \underset{\vec{x} \in \mathcal{U}^N}{\arg\min}\ P(\hat{y_1} \mid \vec{x}) - P(\hat{y_2} \mid \vec{x}), \tag{2.26}$$

kjer so najbolj negotovi tisti primeri, ki imajo najmanjšo razliko med posteriornima verjetnostima $P(\hat{y_1} \mid \vec{x})$ in $P(\hat{y_2} \mid \vec{x})$ dveh najbolj verjetnih razredov y_1 in y_2. Pri tem je dan poudarek razlikovanju med najbolj verjetnim in drugim najbolj verjetnim razredom. Tak pristop tipično zelo dobro deluje v kombinaciji z diskriminatornim algoritmom SVM, ki skuša maksimizirati razdaljo med hiperravninama, ki ločujeta primere obeh razredov. Najbolj negotovi primeri bodo glede na izbrano metriko idealno tisti primeri, ki ležijo točno na robu in jih imenujemo podporni vektorji. Le-ti so za model SVM najbolj informativni.

Drugi primer metrike negotovosti napovedi f je *entropija* (angl. *entropy*):

$$f_{entropija} = \underset{\vec{x} \in \mathcal{U}^N}{\arg\max} -\sum_{i=1}^{k} P(\hat{y_i} \mid \vec{x}) \log P(\hat{y_i} \mid \vec{x}), \tag{2.27}$$

kjer za razliko od metrike f_{rob} ne upoštevamo posteriorne verjetnosti zgolj dveh najbolj verjetnih razredov, temveč posteriorne verjetnosti vseh k razredov. Entropija kot mera nedoločenosti da največjo vrednost primerom, kjer

so si verjetnosti razredov medsebojno kar se da podobne. Z njo upoštevamo negotovost nad celotno verjetnostno distribucijo.

Če pa želimo pri izbiri najbolj informativnih primerov v $\mathcal{U}^N$ upoštevati še korelacijo med primeri, potem izračunamo podobnosti vsakega primera z vsemi ostalimi. Definirajmo enačbo korelacije med primeri

$$g = \underset{\vec{x} \in \mathcal{U}^N}{\arg\max} \frac{1}{m} \sum_{\substack{i=1 \\ \vec{x} \neq \vec{x^{(i)}}}}^{m} S(\vec{x}, \vec{x^{(i)}}), \tag{2.28}$$

kjer m predstavlja število primerov v $\mathcal{U}^N$. Vrednosti korelacije posameznega primera x z vsemi ostalimi primeri v $\mathcal{U}^N$ nato seštejemo in identificiramo tiste, ki so najbolj korelirani oz. povezani z ostalimi. Kot metrika za izračun podobnosti S se tipično uporablja *kosinusna podobnost* (angl. *cosine similarity*), katere pomen je natančno obrazložen v poglavju 2.3.

Metriko g nato kombiniramo z metrikama, ki izbirata primere na podlagi negotovosti napovedi, s čimer pridelamo dve novi kombinirani metriki, ki sta predstavljeni z enačbo:

$$u = \underset{\vec{x} \in \mathcal{U}^N}{\arg\max} \; f \times g^{\gamma} \tag{2.29}$$

Pri tem vrednosti negotovosti izračunane na podlagi bodisi f_{rob} bodisi $f_{entropija}$ pomnožimo s korelacijsko metriko g, katero še dodatno utežimo s parametrom γ. Če je vrednost parametra enaka 1, potem funkcija narašča linearno, kar pomeni, da se vpliv korelacije na kombinirano metriko ne spremeni. Če je $\gamma > 1$, potem dodatno pridobijo na pomenu primeri, ki imajo vrednosti na zgornjem delu intervala med 0 in 1. In obratno, če je $\gamma \leq -1$, odvzamemo pomen primerom z vrednostmi na zgornjem delu intervala med 0 in 1 in ga damo primerom, ki imajo vrednosti na spodnjem delu omenjenega intervala. Z uporabo negativnih vrednosti za parameter γ zmanjšamo vpliv najbolj koreliranih primerov. To je smiselno, ker so le-ti velikokrat medsebojno močno korelirani in z izbiro vseh njih pridobimo bolj malo. Izbira parametra γ je sicer odvisna od domene in narave problema, njena vrednost pa se določi empirično. Kombinacija korelacijskih metrik z metrikama, ki temeljita na negotovosti napovedi, naj bi slednjima ponudila neko dodatno informacijo

pri optimalni izbiri primerov iz $\mathcal{U}^N$ in posledično prinesla večjo napovedno točnost modela.

Celoten postopek aktivnega učenja poteka iterativno. V vsaki iteraciji se izbrani primeri iz $\mathcal{U}^N$ na podlagi ene od opisanih metrik dodajo v $\mathcal{U}^O$, na kateri se nato zgradi nov napovedni model. Sprememba napovedne točnosti se nato preveri nad popolnoma ločeno testno množico $\mathcal{T}$. Tipično se za pridobitev referenčne spodnje meje izboljševanja točnosti modela, vzporedno z enim od opisanih pristopov, izvaja še naključno izbiranje primerov.

2.3 Odkrivanje znanj iz besedil

Odkrivanje znanj iz besedil se nanaša na proces ekstrakcije netrivialnih informacij in znanja iz nestrukturiranega teksta. Področje lahko poimenujemo tudi z izrazom *inteligentna analiza teksta* (angl. *intelligent text analysis*) [6, 11].

Odkrivanje znanj iz besedil je podobno odkrivanju znanj iz podatkov s to razliko, da pri slednjem tipično takoj pridobimo podatke v neki strukturirani obliki (npr. podatki iz urejenih podatkovnih baz, podatki v formatu XML ali JSON itd.). Na drugi strani imamo opravka z nestrukturiranimi ali delno strukturiranimi podatki (npr. HTML dokumenti, elektronska sporočila, nestrukturirani tekstovni dokumenti itd.). To nestrukturirano vsebino je potrebno obdelati, na smiselen način združiti in iz nje ekstrahirati relevantne informacije, ki predstavljajo dodano vrednost. Zaradi tega se odkrivanje znanj iz besedil v veliki meri in z dobrimi rezultati uporablja v velikih podjetjih, ki imajo na voljo kopico dokumentov. Iz njih pridobljeno znanje tako služi kot podpora odločanju in optimizaciji procesov.

Odkrivanja znanj iz besedil se lotimo tako, da tekst sprva predelamo iz nestrukturirane ali delno strukturirane oblike v strukturirano [5]. *Neobdelano besedilo* (angl. *raw text*) moramo torej predelati v obliko, ki bo uporabna za učenje. Med tipične postopke spadajo ekstrahiranje *golih besedil* (angl. *plain text*) iz npr. besedil v formatih HTML. Tem nato odstranimo vsa ločila,

števila in ostale nečrkovne znake, preostalo besedilo pa dodatno spremenimo tako, da so vsi znaki zapisani z malimi črkami.

Tako nam ostane množica besed iz katere za zmanjšanje časovne kompleksnosti obdelovanja besedila in za bolj reprezentativno porazdelitev frekvenc unikatnih besed v besedilu, odstranimo *najbolj pogoste neinformativne besede* (angl. *stop words*).

Nad množico preostalih besed lahko nato izvedemo postopek *krnjenja* (angl. *stemming*) ali pa *lematizacije* (angl. *lemmatization*). Ker krnjenje ne določa pravega korena beseda, namesto slednjih raje govorimo o krnih. Pri krnjenju vsako besedo pretvorimo v njeno krnjeno obliko. S tem močno zmanjšamo problem prevelike dimenzionalnosti, saj številne predhodno različne, a vendar v svoji osnovi zelo podobne termine združimo v eno samo entiteto. Slednje velja tudi za lematizacijo, vendar s to razliko, da je krnjenje bolj grob postopek. Pri lematizaciji ne upoštevamo krna besede, temveč poskušamo najti morfološko lemo besede. Prislikajmo razliko med krnjenjem in lematizacijo na preprostem primeru dveh besed: "človek" in "ljudje". Postopek krnjenja bi vzel krna obeh besed in ju na koncu označil kot različni entiteti, lematizacija pa bi prepoznala, da je morfološka lema obeh besed enaka in ju združila v enotno entiteto.

Za dejansko ekstrakcijo znanja iz besedila se poslužujemo metod strojnega učenja [11, 5]. Učni algoritmi tipično sprejemajo vhodne podatke v numerični obliki, zato se pri obdelavi naravnih jezikov in informacijskem poizvedovanju pogosto uporablja posebna predstavitev besedila v obliki modela vreča besed. Pri slednjem predstavimo korpus besedil, nad katerim skušamo zgraditi model, z matriko številnosti. V njej stolpci (značilke) predstavljajo unikatne besede, ki se pojavijo vsaj enkrat v kateremkoli izmed besedil, vrstice pa posamezne primere besedil. Vsakemu besedilu torej pripada vektor, ki vsebuje številnosti oz. *frekvence besed* (angl. *term frequencies*), ki se pojavljajo v njem. Ta matrika ima posebno ime, t.j. *matrika dokument-beseda* (angl. *document-term matrix*). Reprezentacija podatkov, kjer vsaka nova unikatna beseda doda eno dimenzijo matriki dokument-beseda, prinese visoke časovne

kompleksnosti pri dejanskem izvajanju algoritmov oz. pri gradnji napovednih modelov, zato se v praksi uporablja predstavitev z redkimi matrikami, ki močno izboljšajo performanse metod.

Poglejmo si primer, kjer imamo dve besedili: (1) "Žiga rad gleda filme. Tudi Tina ima rada filme." in (2) "Žiga rad gleda tudi nogometne tekme." V obeh besedilih poiščemo skupne unikatne besede, ki so "Žiga", "Tina", "rad", "gledati", "imeti", "film", "tudi", "nogomet", "tekma", torej vsega skupaj devet. Seznam unikatnih besed ustreza devetih stolpcem, medtem ko vsaka vrstica predstavlja frekvence besed v posameznem besedilu. Matrika dokument-beseda ima torej sledeče vrednosti:

$$\begin{array}{c} \\ (1) \\ (2) \end{array} \begin{array}{c} \begin{array}{ccccccccc} \check{Ziga} & Tina & rad & gledati & imeti & film & tudi & nogomet & tekma \end{array} \\ \begin{pmatrix} 1 & 1 & 2 & 1 & 1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array} \tag{2.30}$$

V klasičnih oblikih matrika dokument-beseda vsebuje zgolj frekvence besed po besedilih. Pri klasifikaciji besedil in informacijskem poizvedovanju pa je značilno, da to osnovno vsebino utežimo na podlagi določene sheme. Med najbolj znanimi je utežitev TF-IDF:

$$TF\text{-}IDF(t,d,D) = TF(t,d) \times IDF(t,D) = TF(t,d) \times \log \frac{N}{n_t}, \tag{2.31}$$

kjer $TF(t,d)$ predstavlja frekvenco besede t v besedilu d, $IDF(t,D)$ relativno redkost besede t v celotnem korpusu besedil D (N predstavlja število vseh besedil v korpusu, n_t pa število besedil, ki vsebujejo besedo t), $TF\text{-}IDF(t,d,D)$ pa *relativno redkost besede* (angl. *inverse document frequency*) t v besedilu d uteženo z relativno redkostjo besede d v celotnem korpusu besedil D. Z utežitvijo TF-IDF poleg frekvenc besed v besedilu upoštevamo tudi relativno redkost besed v korpusu besedil, t.j. množici vseh dokumentov, ki jih analiziramo oz. nad katerimi delamo poizvedbo. Ta utežitev skuša prikazati, kako pomembna je določena beseda, ki se pojavlja v dokumentu v kontekstu celotnega korpusa. Vrednost TF-IDF se spreminja premo sorazmerno s frekvenco besede v dokumentu in obratno sorazmerno s frekvenco besede v celotnem

korpusu. Opisan postopek nam pomaga pri ublažitvi vpliva besed, ki se v splošnem v vseh besedilih pojavljajo bolj pogosto kot druge, saj iz teorije informacij vemo, da bolj pogoste besede nosijo manj informacije.

Na podlagi matrike dokument-beseda izračunamo podobnost med besedili, za ta namen pa tipično uporabljamo mero imenovano kosinusna podobnost [6, 7, 11]:

$$S(\vec{a},\ \vec{b}) = \frac{\vec{a}^{\,T}\vec{b}}{\|\vec{a}\|\|\vec{b}\|} \tag{2.32}$$

Za par besedil dobimo medsebojno kosinusno podobnost S tako, da izračunamo skalarni produkt med vektorjema $\vec{a}$ in $\vec{b}$, to vrednost pa delimo s produktom dolžin obeh vektorjev. Mera je zelo uporabna na področju odkrivanja znanj iz besedil, saj pri izračunu, z delitvijo skalarnega produkta s produktom dolžin, odpravimo problem različnih dolžin besedil . Tako velja, da bolj kot sta si besedili podobni, manjši je kot med pripadajočima vektorjema, in obratno.

Poglavje 3

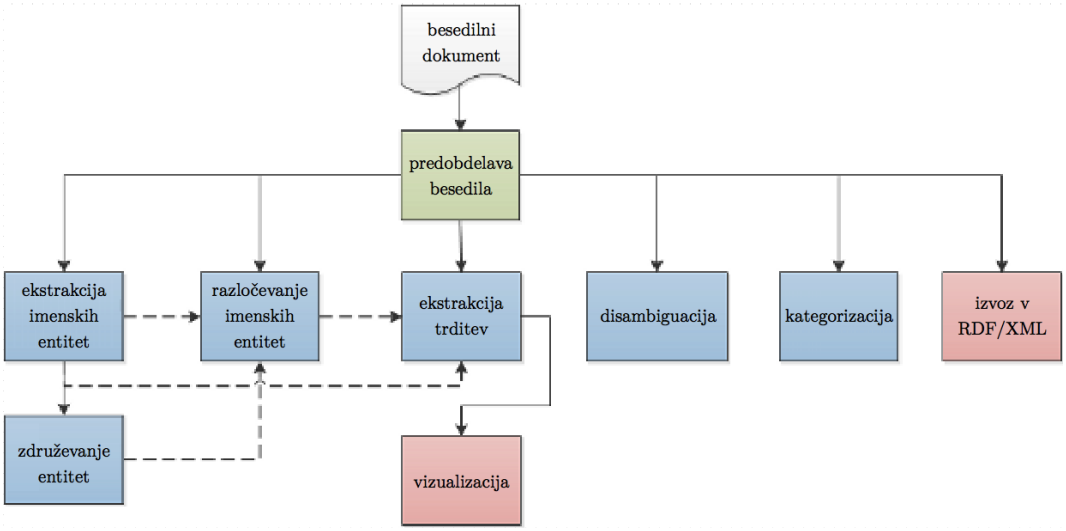
Uporabljene tehnologije in orodja

3.1 Enrycher, IJS Newsfeed in Event Registry

Enrycher[1], IJS Newsfeed[2] in Event Registry[3] so sistemi, ki so bili razviti v Laboratoriju za umetno inteligenco na Institutu Jožef Stefan [27, 28, 30, 9, 10]. Med njimi obstaja povezava, in sicer velja, da sistem Enrycher semantično obogati besedila za sistem IJS Newsfeed, slednji pa predstavlja vir podatkov (člankov) sistemu Event Registry.

Enrycher je sistem, ki omogoča zgolj površinsko ali pa bolj poglobljeno procesiranje besedil na nivoju dokumentov. Na sliki 3.1 je prikazana shema njegovega delovanja. V primeru površinske obdelave besedil se izvedejo prepoznavanje *teme* (angl. *topic*) besedila, iskanje ključnih besed v besedilu in *ekstrakcija imenskih entitet* (angl. *named entity extraction*), kot so npr. imena ljudi, lokacij in organizacij ter datumi, odstotki in denarni zneski. V primeru poglobljenega procesiranja besedil pa se izvede prepoznavanje imenskih entitet s pomočjo baz podatkov DBpedia[4], YAGO[5] in OpenCyc[6], di-

[1] `http://Enrycher.ijs.si`
[2] `http://Newsfeed.ijs.si`
[3] `http://eventregistry.org`
[4] `http://wiki.dbpedia.org`
[5] `http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago//`
[6] `http://sw.opencyc.org`



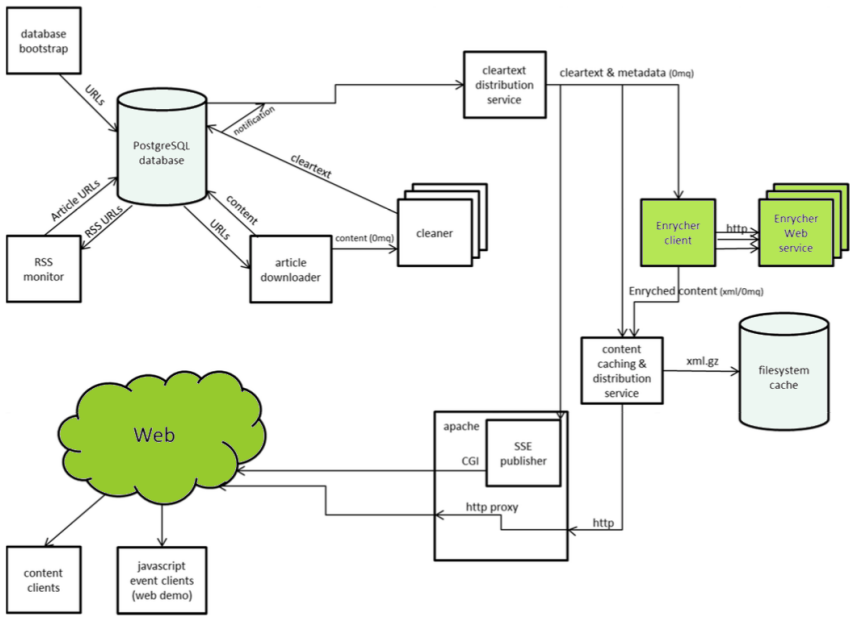
Slika 3.1: Shematski prikaz delovanja sistema za procesiranje besedil Enrycher. Slika povzeta po [26].

sambiguacija besed z uporabo semantičnega leksikona angleškega jezika WordNet[7] ter ekstrakcija trditev preko prepoznavanja stavčnih elementov oblike osebek - povedek - predmet. Spletni programski vmesnik, ki izpostavlja sistem vrača vsebino bodisi v obliki RDF bodisi v obliki XML. Zaenkrat ima sistem podporo le za angleški in slovenski jezik. Kot že omenjeno, funkcionalnosti sistema Enrycher uporablja sistem IJS Newsfeed.

IJS Newsfeed je sistem, ki preko spletnega programskega vmesnika izpostavlja zvezen, v realnem času agregiran tok semantično obogatenih novic pridobljenih s pomočjo pregledovanja RSS spletnih strani po celem svetu. Sistem dnevno zajame od 100 do 150 tisoč člankov s pomočjo 75 tisoč RSS spletnih strani. Ima podporo za 35 jezikov, pri čimer je največ (50 %) zajetih člankov v angleškem jeziku, sledi nemščina z 10 %, nato španščina z 8 %, itd. Na sliki 3.4 je prikazana arhitektura sistema, njegovo delovanje pa lahko opišemo v naslednjih petih korakih:

1. Periodično *prehodi* (angl. *crawl*) seznam RSS spletnih strani in pridobi

[7] `https://wordnet.princeton.edu`



Slika 3.2: Arhitektura sistema IJS Newsfeed. Slika povzeta po [30]. (V izogib nejasnostim na sliki navajamo angleške termine komponent sistema.)

povezave do novičarskih člankov.

2. Prenese HTML vsebino spletne strani in pri tem pazi, da ne obremeni gostujočih strežnikov.

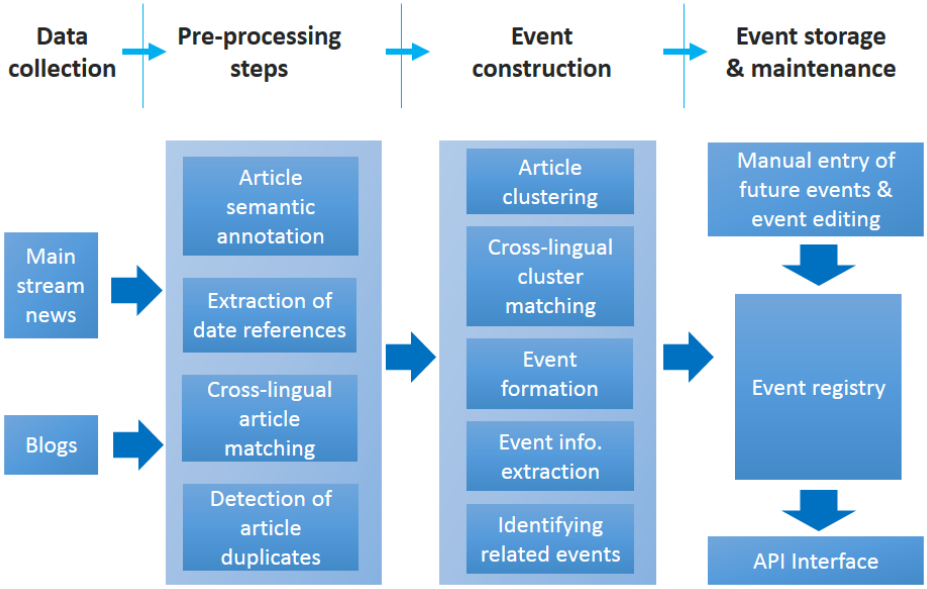
3. Razčleni vsebino pozameznega članka in pridobi potencialne nove RSS vire (ki bodo uporabljeni v novi iteraciji iskanja člankov v koraku 1.) ter očiščeno besedilo jedra članka.

4. Obdela članke s sistemom Enrycher in pridobi meta-podatke.

5. Preko spletnega programskega vmesnika izpostavi dva toka novičarskih člankov (očiščeno besedilo in besedilo obogateno z meta-podatki).

Spletni vmesnik sistema IJS Newsfeed nam vrača kronološko urejene novičarske članke v formatu XML.

Event Registry je sistem, ki v realnem času zbira in analizira globalno objavljene spletne novice. Pri tem identificira skupine novičarskih člankov v več



Slika 3.3: Arhitektura sistema Event Registry. Slika povzeta po [10]. (V izogib nejasnostim na sliki navajamo angleške termine komponent sistema.)

jezikih, ki opisujejo isti svetovni dogodek. Po tem ključu jih grupira, nato pa iz njih samodejno ekstrahira pomembne informacije (lokacija dogodka, datum, udeleženci in o čem dogodek govori), jih shrani in smisleno izpostavi končnim uporabnikom na spletu preko spletnega programskega ali uporabniškega vmesnika. Medtem ko oba omogočata poizvedbe po dogodkih s široko paleto iskalnih parametrov, spletni uporabniški vmesnik omogoča še vizualizacijo in agregacijo rezultatov iskanja za lažje seznanjanje s pozameznim dogodkom in kot pomoč pri identifikaciji povezanih dogodkov. Arhitektura sistema ima 4 glavne korake, ki si zaporedno sledijo:

Pridobitev podatkov. Glavni podatkovni vir je sistem IJS Newsfeed.

Predhodna obdelava podatkov. V tem koraku se izvede semantična obogatitev člankov z uporabo algoritma za prepoznavanje imenskih entitet, ki ima podprto disambiguacijo. Obenem sistem ekstrahira pomembne informacije o posameznem članku, odkrije duplikate in poskuša iden-

tificirati podobne članke v različnih jezikih (trenutno lahko prepozna podobnost med članki v štirinajstih različnih jezikih, in sicer angleškem, nemškem, španskem, katalonskem, portugalskem, italijanskem, francoskem, ruskem, arabskem, turškem, kitajskem, slovenskem, hrvaškem in srbskem).

Gradnja dogodkov. V tej fazi sistem poišče skupine člankov, ki opisujejo isti dogodek in iz njih ekstrahira informacije o dogodku. Ta postopek se izvaja s pomočjo gručenja, pri čimer se za vsakega od štirih podprtih jezikov zgradi ločen model. Rezultat gručenja so skupine člankov, ki opisujejo isti dogodek v enem od podprtih jezikov. Za združevanje skupin istih dogodkov v različnih jezikih v eno samo, sistem zgradi model z algoritmom SVM. Na podlagi identificiranih skupin oz. gruč se ustvarijo novi dogodki. Vsakemu dogodku se priredi še DMoz[8] kategorija, ki je osnovana na hierarhični taksonomiji ročno kuriranega portala, ki organizira pet milijonov spletnih strani v skupine, pri čimer sistem uporabi le najvišje tri nivoje omenjene taksonomije. Sistem klasificira dogodek preko podobnosti vsebine člankov zaznanega dogodka z vsebino člankov posamezne DMoz kategorije.

Hranjenje dogodkov in vzdrževanje. Na koncu so identificirani dogodki z vsemi pridobljenimi informacijami shranjeni v podatkovno bazo. Do njih lahko dostopajo uporabniki preko spletnega vmesnika.

3.2 Portal dogodkov Wikipedia

V okviru spletne enciklopedije Wikipedia obstaja portal dogodkov[9], kjer uporabniki ročno uvrščajo članke iz različnih virov v kategorije po dnevih. Portal vsebuje članke vse nazaj do leta 1999, vendar so šele od sredine leta 2010 vsi navedeni članki oz. reference na njih klasificirane v eno od kategorij. Taksonomija je nehierarhične oblike, opredeljena pa je z malo več kot desetimi

[8] `http://www.dmoz.org`

[9] `https://en.wikipedia.org/wiki/Portal:Current_events`

kategorijami. Nekatere izmed njih so nereprezentativno zastopane, zato lahko za potrebe klasifikacije uporabimo le članke osmih kategorij. Ker portal ne ponuja spletnega programskega vmesnika, je edina možnost za pridobitev referenc na članke *strganje* (angl. *scraping*) HTML vsebine spletne strani in obdelava neobdelanega besedila.

3.3 Guardian API

Britanska časopisna hiša The Guardian vsebino, ki jo proizvaja, ponuja v elektronski obliki preko spletnega programskega vmesnika[10]. Slednji ponuja dostop do preko enega in pol milijona člankov in ostalih novičarskih vsebin, ki datirajo nazaj vse do leta 1999. Gradivo je urejeno po sekcijah, besedilo člankov pa obogateno z meta-podatki v obliki ključnih besed.

Spletni programski vmesnik je preprost za uporabo, a hkrati ponuja dovolj globine pri kreiranju poizvedbe. Tako lahko filtriramo iskanje po vsebini besedila. Primer vsebinskega filtra bi bil npr. “war AND (terrorism OR attack)”. V vsebinskem filtru je torej omogočeno povezovanje besed z logičnimi vezniki, kar nam olajša poizvedovanje in pomaga pri iskanju člankov točno določene kategorije. Uporabimo lahko tudi druge filtre, kot so sekcija (npr. “world”, “culture”, “business”, “sport”...), datumske omejitve itd. Tudi vsi ostali filtri omogačajo povezovanje besed z logičnimi vezniki.

Poizvedba nam vrne objekte v formatu JSON, ki med drugim vsebujejo naslov in vsebino članka, ključne besede, URL članka itd. Ker sami natančno specificiramo poizvedbo, lahko pridobljene članke avtomatsko uvrstimo v določeno predhodno definirano kategorijo. Skratka Guardian API je odličen vir za pridobivanje kvalitetne tekstovne vsebine v obliki časopisnih člankov, ki je dodatno obogatena z meta-podatki.

[10] `http://open-platform.theguardian.com`

3.4 Sistem NELL

NELL[11] je sistem, ki se neprestano uči s spleta. Temelji na metodah strojnega učenja, razvijajo pa ga na Univerzi Carnegie Mellon v Združenih državah Amerike v okviru projekta "Read the Web". Med neprekinjenim iterativnim delovanjem, ki traja že več kot pet let, NELL vsak dan prebere določene spletne tekstovne vsebine, iz njih ekstrahira dejstva oz. fakte ter nato samoizboljšuje lastne sposobnosti branja, kar omogoči bolj natančno branje teksta v naslednji iteraciji. Do sedaj je NELL pridobil že preko osemdeset milijonov omenjenih dejstev, katerim je pripisana določena stopnja zaupanja. Sistem ni popoln, saj ima visoko stopnjo zaupanja le okoli dva in pol milijona dejstev, vendar se z neprestanim učenjem izboljšuje in veča svojo *bazo znanja* (angl. *knowledge base*).

Cilj sistema NELL je posnemanje ljudi pri učenju, pri čimer slednji zavzame vlogo *agenta, ki se neprekinjeno uči* (angl. *Never-Ending Learning Agent*) in se kot tak sooča z množico učnih problemov. Agent je sistem, ki tako kot človek tekom življenja pridobi veliko različnih tipov znanja, le-ti pa temeljijo na različnih, samonadzorovanih izkušnjah. Z neprestanim povečevanjem baze znanja izboljšuje sposobnost učenja v prihodnosti. Prav tako skuša posnemati človekovo sposobnost samorefleksije in oblikovanja novih oblik predstavitev znanja. Ta lastnost človeku omogoča, da najde novo inovativno pot do rešitve, agentu pa pomaga pri izogibanju stagnaciji v smislu učinkovitosti tekom procesa učenja.

Tako lahko definiramo *neprestano prisotni učni problem* (angl. *Never-Ending Learning Problem*):

Neprestano prisotni učni problem $\mathcal{L}$, s katerim se sooča agent A, sestoji iz množice učnih nalog L in množice povezav med rešitvami učnih nalog C:

$$\mathcal{L} = (L, C) \tag{3.1}$$

L_i predstavlja i-ti učni problem, čigar cilj je izboljšati agentov učinek P_i pri

[11] `http://rtw.ml.cmu.edu/rtw/`

reševanju naloge T_i ob danem naboru izkušenj E_i:

$$L_i = < T_i, P_i, E_i > \tag{3.2}$$

C predstavlja množico sklapljajočih povezav med rešitvami učnih problemov, ϕ_k je realna funkcija, ki podaja stopnjo povezave za minimalno dva učna problema, V_k pa je vektor indeksov, ki določa katere učne naloge so del dane povezave:

$$C = < \phi_k, V_k > \tag{3.3}$$

T_i predstavlja i-to učno nalogo agenta, par $< X_i, Y_i >$ pa opisuje domeno in obseg naučene funkcije f_i, ki pripada T_i:

$$T_i \equiv < X_i, Y_i > \tag{3.4}$$

$$f_i : X_i \to Y_i \tag{3.5}$$

P_i predstavlja metriko za merjenje učinka naučene funkcije f_i. Z njo definiramo optimalno naučeno funkcijo f_i^* za T_i. F_i predstavlja prostor vseh možnih preslikav med X_i in Y_i:

$$P_i : f_i \to \mathbb{R} \tag{3.6}$$

$$f_i^* \equiv \underset{f_i \in F_i}{\arg\max}\ P_i(f_i) \tag{3.7}$$

Učni problem sestoji iz n učnih nalog, torej agent A pridela množico rešitev v obliki optimalnih naučenih funkcij številnosti n, katerih kvaliteta se s časom postopoma izboljšuje. Obenem velja, da različnih učnih nalog ni potrebno reševati hkrati in da rešitev ene naloge pripomore k rešitvi naslednjih.

Aplikacija koncepta agenta, ki se neprekinjeno uči v obliki sistema NELL:
Vhod:

- Osnovna ontlogija, ki definira tipe entitet (npr. "(Sport, Athlete)") in binarne relacije (npr. "AthletePlaysSport(Jordan, Basketball)").
- Okoli 12 označenih učnih primerov za vsak tip entitete in binarno relacijo.

- 500 milijonov spletnih strani ter dostop do 100.000 iskanj na dan preko iskalnika Google.
- Občasna interakcija z ljudmi, ki preko procesa aktivnega učenja izboljšuje sposobnost branja spleta.

Ponavljaj:

1. Ekstrahiraj nova dejstva iz prebranih strani s spleta in izbriši stara nepravilna dejstva. S tem dopolnjuj bazo znanja. Za vsako dejstvo hrani *zaupanje* (angl. *confidence*) in *izvor* (angl. *provenance*).
2. Z učenjem novih modelov oz. funkcij izboljšaj sposobnost branja spletnih strani.

   Neprekinjeno, 24 ur na dan

NELL prav tako poseduje sposobnost, da iz že pridobljenih dejstev shranjenih v bazi znanja, sklepa oz. povzame nova dejstva, ki še niso bila prebrana. S tem sistem sam avtomatsko obogati svojo začetno, ročno podano ontologijo.

Sistem se sooča z 2.500 učnimi problemi, ki sestojijo iz nalog T_i, metrike za merjenje učinka P_i in tipa izkušenj E_i. V sistemu NELL predstavlja metrika P_i, ki jo slednji skuša optimizirati pri reševanju T_i, zgolj točnost naučene funkcije oz. naučenega napovednega modela. Tip izkušenj E_i pa je kombinacija začetnih, ročno označenih učnih primerov, množice avtomatsko označenih učnih primerov v bazi znanja ter neoznačenega besedila iz spletnih strani.

Poznamo več tipov učnih nalog, in sicer klasifikacijo v tipe entitet, klasifikacijo v relacije, odkrivanje entitet ter pravila za sklepanje med trojčki dejstev.

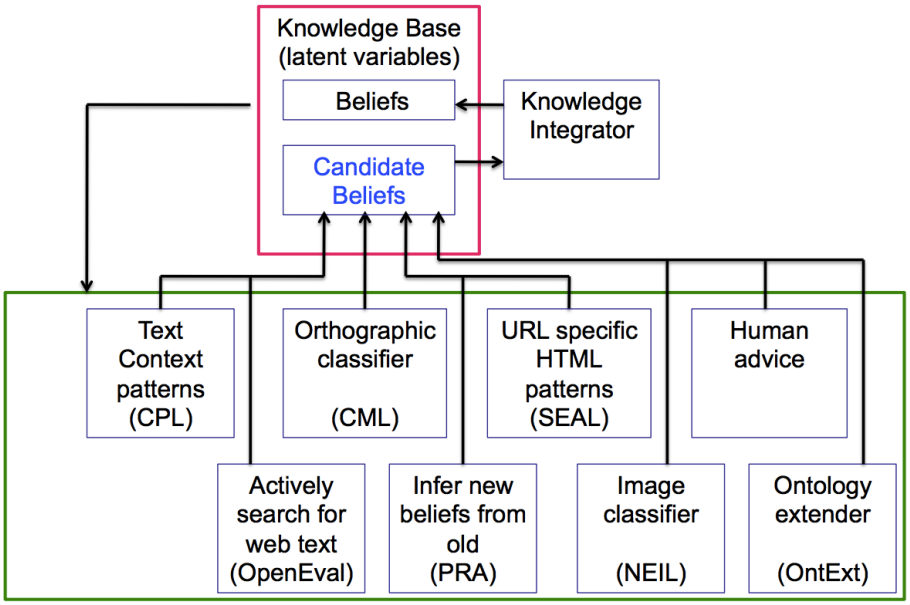
Pri klasifikaciji v tipe entitet velja, da sistem skuša uvrstiti *samostalniške fraze* (angl. *noun phrases*) v semantične tipe entitet. Vseh tipov entitet je 280, posamezna samostalniška fraza pa lahko pripada več tipom entitet hkrati. Za napovedovanje posameznega tipa entitet Y_i se NELL nauči do pet

različnih modelov. Vsak izmed petih tipov modelov ima za učenje na voljo drugačen atributni prostor X_i, ki opisuje samostalniško frazo: (1) besedilo v obliki vektorja s tekstovnimi atributi, (2) porazdelitev konteksta besedila okoli najdenih primerkov samostalniške fraze v korpusu 500 milijonov spletnih strani, (3) porazdelitev konteksta besedila okoli najdenih primerkov samostalniške fraze v besedilih pridobljenih s spletnim iskanjem, (4) HTML struktura spletnih strani, ki vsebuje samostalniško frazo in (5) slike povezane s samostalniško frazo.

Pri klasifikaciji v relacije, NELL klasificira par samostalniških fraz glede na to, ali frazi ustrezata izbrani relaciji. Primer: imamo par samostalniških fraz "<New York, USA>", za katerega moramo določiti, ali ustreza relaciji "CityLocatedInCountry(x, y)". Sistem v svoji ontologiji trenutno vsebuje 327 relacij Y_i, za vsako izmed njih pa zgradi 3 različne klasifikacijske modele z različnimi atributnimi prostori X_i, ki opisujejo par samostalniških fraz: (1) porazdelitev konteksta besedila med najdenimi primerki para samostalniških fraz v korpusu 500 milijonov spletnih strani, (2) porazdelitev konteksta besedila med najdenimi primerki para samostalniških fraz v besedilih pridobljenih s spletnim iskanjem in (3) HTML struktura spletnih strani, ki vsebuje obe samostalniški frazi, ki tvorita omenjeni par.

Učne naloge za odkrivanje entitet iščejo sinonime med pari. Na primer, da imamo samostalniški frazi "New York" in "Big Apple", za njiju pa želimo napovedati, ali sta oba del iste entitete (ali sta sinonima). NELL se s tem problemom spopade tako, da za vsako izmed 280 tipov entitet zgradi dva klasifikatorja, ki imata različna atributna prostora: (1) *tekstovna podobnost* (angl. *string similarity*) med samostalniškima frazama in (2) podobnost med njunimi ekstrahiranimi dejstvi.

Pravila za sklepanje med trojčki dejstev kot zadnja skupina učnih nalog sistema NELL, poskušajo zgraditi modele, ki bi znali kar se da dobro preslikati trenutna dejstva v bazi znanja v dejstva, ki bi morala biti tja dodana v prihodnje. Za vsako relacijo v ontologiji je zgrajen model, ki vsebuje zbirko naučenih omejenih pravil *Hornovih klavzul* (angl. *Horn clauses*). Hornova



Slika 3.4: Arhitektura sistema NELL. Slika povzeta po [12]. (V izogib nejasnostim na sliki navajamo angleške termine komponent sistema.)

klavzula je logična formula, ki sestoji iz disjunktno povezanih *členov* (angl. *literals*) in vsebuje največ en pozitivni člen. Če vsebuje natanko en pozitivni člen, potem govorimo o fiksni oz. ekzaktni klavzuli, če ne vsebuje nobenega negativnega člena, potem to predstavlja dejstvo, v primeru da ni prisotnega nobenega pozitivnega člena, pa dobimo ciljno klavzulo.

Učenje v sistemu NELL predstavlja približek algoritmu EM, saj se tekom izvajanja delno nadzorovanega učenja v vsaki iteraciji sprva izvede korak pričakovanja E, nato pa korak maksimizacije M. V koraku E sistem z ekstrahiranimi dejstvi, katerim pripadajo zaupanje in izvor, predlaga posodobitve v bazi znanja. Končno odločitev o dodajanju oz. brisanju nekega dejstva izvrši t.i. *integrator znanja* (angl. *knowledge integrator*). Slednji odloča na podlagi vrednosti zaupanja, zaradi želje po manjši časovni in prostorski kompleksnosti pa se ohrani le tiste, ki imajo visoko vrednost. V koraku M se na podlagi posodobljene baze znanja ponovno zgradijo vsi modeli, ki jih NELL

uporablja pri učnih nalogah. Rezultat učenja vseh teh modelov je sistem, v katerem rezutati več tisoč učnih nalog medsebojno vplivajo na učenje.

Sistem NELL omogoča dostop do baze znanja preko spletnega programskega vmesnika, ki vrača objekte v formatu JSON. Vmesnik omogoča dva tipa poizvedovanj, in sicer prvega v obliki *trditev* (angl. *assertions*), drugega pa z uporabo *nadomestnih znakov* (angl. *wildcard*). Pri prvem podamo bodisi par "(parameter, tip entitete)" bodisi trojček "(prvi parameter, relacija, drugi parameter)", vmesnik pa nam vrne enega ali več rezultatov, ki povedo, s kakšno verjetnostjo lahko trdimo, da podana trditev velja. Pri drugem primeru pa lahko namesto tipa entitete, relacije ali drugega parametra podamo nadomestni znak $*$, vmesnik pa nam odgovori z množico dejstev in njim pripadajočim vrednostim zaupanja, ki ustrezajo dani poizvedbi [12, 2].

3.5 Programska orodja

Večina programskega dela rešitve je bila napisana v programskem jeziku Python [23], ki ima močno podporo za implementacijo rešitev problemov strojnega učenja. Programski knjižnici NumPy [16] in SciPy [18] ponujata širok nabor podatkovnih struktur in funkcij, ki omogočajo učinkovito izvajanje numeričnih operacij. Knjižnica scikit-learn [17] je odprtokodno orodje, namenjeno specifično strojnemu učenju, ki vsebuje zelo učinkovite implementacije vseh omenjenih učnih algoritmov. Za strganje HTML vsebine s spletnih strani smo uporabili knjižnico scrapy [19], za vizualizacijo pa knjižnico matplotlib [15]. V okviru predhodne obdelave podatkov smo za krnjenje uporabili Porter Stemmer [24], ki je del knjižnice NLTK. Za potrebe shranjevanja in nalaganja podatkov, predhodnega procesiranja, internega prečnega preverjanja in evalvacije rezultatov učnih algoritmov, je bila uporabljena lastna knjižnica imenovana mllib.

Del rešitve, ki je namenjen ročnemu označevanju člankov pridobljenih s spletnega programskega vmesnika IJS Newsfeed, je bil implementiran v programskih jezikih JavaScript [20], jQuery [21] in PHP [22].

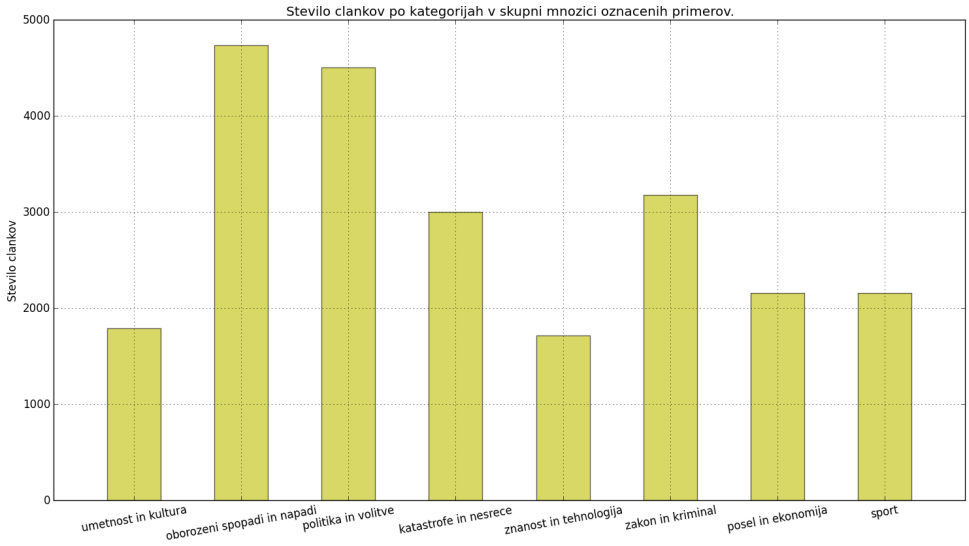
Poglavje 4

Napovedovanje kategorij novičarskih člankov

4.1 Članki in meta-podatki

Podatki v obliki označenih časopisnih člankov so bili pridobljeni s portala dogodkov spletne enciklopedije Wikipedia ter preko uporabe spletnega programskega vmesnika britanske časopisne hiše The Guardian. Neoznačeni članki so bili zajeti preko spletnega programskega vmesnika portala IJS Newsfeed. Dodatni meta-podatki (ključne besede, entitete in kategorije) pa smo pridobili (1) s pomočjo spletnega programskega vmesnika sistema NELL, ki izpostavlja lastno bazo znanja, (2) s pomočjo spletnega programskega vmesnika sistema Enrycher in (3) preko spletnega programskega vmesnika sistema Event Registry.

Število označenih člankov pridobljenih s portala Wikipedia je znašalo 14.581, število člankov pridobljenih s pomočjo spletnega programskega vmesnika The Guardian pa 8.649. Skupno število označenih člankov je bilo tako enako 23.230. Število neoznačenih člankov, ki smo jih pridobili s spletnega programskega vmesnika sistema IJS Newsfeed, je bilo 56.850. Označeni članki so bili klasificirani v kategorije glede na taksonomijo portala dogodkov Wikipedia. Taksonomija je bila nehierarhično oblikovana in je kot taka

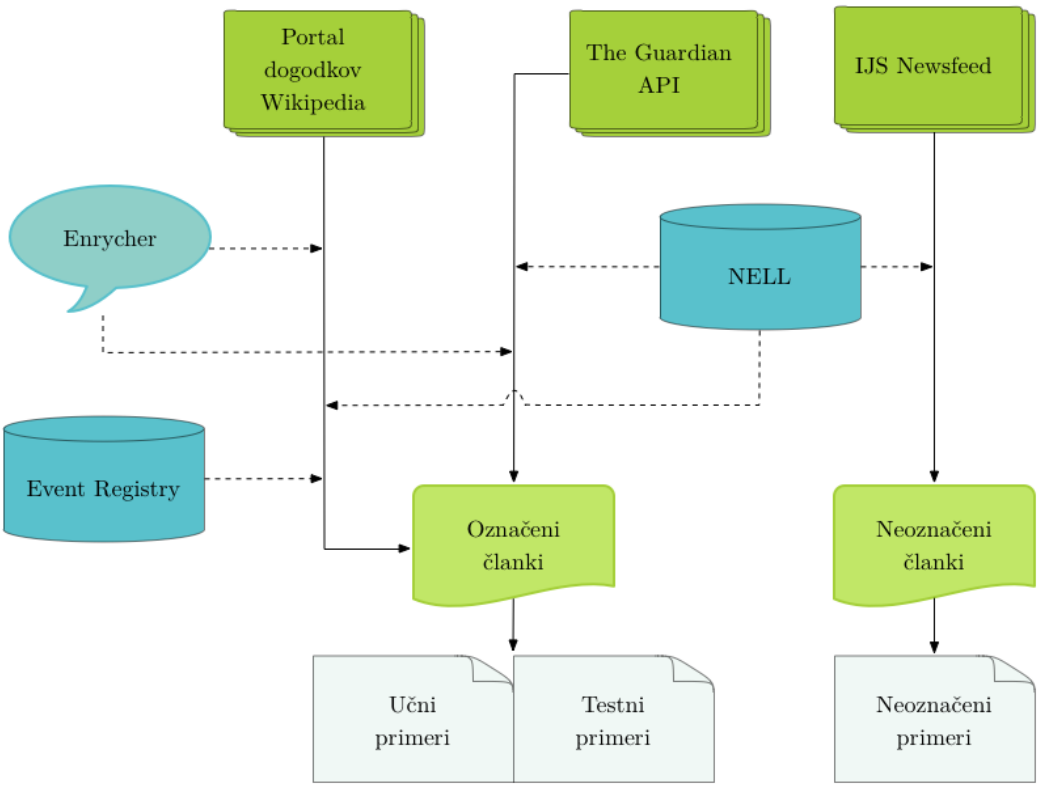


Slika 4.1: Število člankov po kategorijah v skupni množici označenih primerov.

vsebovala zgolj en nivo. Kategorije so bile "oboroženi spopadi in napadi", "umetnost in kultura", "posel in ekonomija", "katastrofe in nesreče", "zakon in kriminal", "politika in volitve", "znanost in tehnologija" in "šport". Slika 4.1 prikazuje število vseh označenih člankov po kategorijah. Tipično so v problemih strojnega učenja neenakomerne zastopanosti razredov velik problem, vendar je v našem primeru množica označenih primerov zelo velika, zato neenakomerne številnosti razredov ne nosijo velike teže.

Za lažje razumevanje procesa pridobivanja podatkov in meta-podatkov je na sliki 4.2 prikazana shema postopka.

Članke Wikipedia smo pridobili tako, da smo s portala dogodkov postrgali HTML vsebino spletne strani in izluščili URL naslove člankov in njihove kategorije. Nato smo s pomočjo spletnega programskega vmesnika sistema Event Registry, ki omogoča poizvedovanje po člankih z URL naslovi, pridobili naslove, besedila in DMoz kategorije za vsak članek posebej. (DMoz kategorija in kategorija sta različna pojma). Za pridobitev imenskih entitet (v nadaljevanju entitet) smo uporabili spletni programski vmesnik sistema



Slika 4.2: Shema, ki prikazuje proces pridobivanja podatkov in meta-podatkov.

Enrycher, ki je v povprečju vrnil 22,2 entitet na članek. Iz DMoz kategorij smo ključne besede pridelali po sledečem postopku: DMoz kategorije vseh člankov, ki so bile povezane v večnivojsko hierarhijo (npr. "health/public health and safety/emergency services", "business/energy/utilities"...) smo razčlenili na atomarne enote ("health", "public health and safety", "emergency services", "business", "energy", "utilities",...) in nad njimi zgradili model vreča besed z uporabo TF-IDF utežitvene sheme. Za posamezni članek smo izluščili ključne besede tako, da smo izmed vseh atomarnih enot članka vzeli tiste, katerih vrednost je presegla empirično določeno konstanto. Na koncu smo na članek v povprečju pridelali 5,1 ključnih besed.

Članki The Guardian so bili pridobljeni s pomočjo spletnega program-

skega vmesnika, in sicer za vsako kategorijo okoli 1.100. Spletni programski vmesnik The Guardian je v odgovorih na naše poizvedbe vračal naslove, gola besedila, ključne besede ter URL-je člankov. Ključnih besed na članek je bilo v povprečju 4,1. Za vsak članek smo želeli pridobiti še entitete, zato smo gola besedila preprocesirali s sistemom Enrycher. Ker so bila besedila člankov iz množice The Guardian dajša kot tista iz množice Wikipedia, je bilo tu povprečno število pridobljenih entitet 33,4 na članek.

Neoznačeni članki, ki smo jih pridobili preko spletnega programskega vmesnika sistema IJS Newsfeed so vsebovali naslov in besedilo, obenem pa so imeli vključene tudi ključne besede in entitete, saj so bili v okviru sistema IJS Newsfeed že obogateni s pomočjo sistema Enrycher. Ključnih besed je bilo na članek v povprečju 4,2, entitet pa 5,7, kar je občutno manj kot pri člankih iz množice Wikipedia in The Guardian.

Za vse skupine člankov, označene in neoznačene smo pridobili še tipe entitet iz baze znanja sistema NELL preko spletnega programskega vmesnika. Ključ poizvedbe so bile ključne besede in entitete posameznega članka. Baza znanja sistema NELL ni popolna in kot taka ne vsebuje vseh poljubnih pojmov, zato so bile nekatere poizvedbe neuspešne. Natančneje, uspešna je bila le približno vsaka enaintrideseta poizvedba. V okviru lastne rešitve smo si lokalno zgradili slovar, ki za vsako ključno besedo ali entiteto, ki jo vsebuje, vrača množico tipov entitet in zaupanj (realno število med 0 in 1), ki jih ima v to dejstvo oz. trditev sistem NELL oz. njegova baza znanja.

4.2 Klasifikacijski modeli

Naš cilj je bil zgraditi klasifikacijski model, ki bi imel kar se da visoko točnost napovedovanja kategorij novičarskih člankov. Referenčni model, ki smo ga želeli izboljšati je podrobno opisan v članku [8]. Slednji je bil zgrajen na podlagi 13.883 označenih člankov pridobljenih s portala dogodkov Wikipedia, testiran pa na 8.168.745 neoznačenih člankih. Uporabljeni sta bili dve skupini atributov, in sicer besedni n-grami velikosti 1, t.j. unigrami ter

meta-podatki v obliki imenskih entitet in ključnih besed. Kot učni algoritem je bil uporabljen SVM.

Pri naši rešitvi smo na podlagi označenih člankov gradili in testirali modele z različnimi meta-podatki, različnimi algoritmi in parametri. Da smo to lahko storili, smo očistili naslove in besedila vseh člankov, tako da smo izločili vsa ločila, števila in ostale posebne znake, nato pa pretvorili besedila v male črke. Sledilo je izločanje najbolj pogostih neinformativnih besed ter krnjenje. Na podlagi preostalih besed v besedilih smo nato zgradili model vreča besed, čigar produkt je bila matrika dokument-beseda. Značilke pridelane na ta način so predstavljale unigrame.

Na podlagi meta-podatkov smo prav tako zgradili modele vreča besed. Entitete in ključne besede člankov so bile predstavljene kot binarne značilke, NELL-ove tipe entitet pa smo uporabili na dva načina, in sicer (1) kot v primeru unigramov so značilke predstavljale frekvenco pojavitev določenega tipa entitete v posameznem članku in (2) namesto, da smo za vsako pojavitev tipa entitete v članku prišteli 1, smo seštevali zanesljivosti oz. verjetnosti pozameznih tipov entitet. S temi značilkami smo močno obogatili atributni prostor.

Matrike dokument-beseda smo utežili s shemo TF-IDF v dveh delih. V prvem delu smo skupaj normalizirali unigrame in NELL-ove tipe entitet, saj so značilke v njunih matrikah temeljile na frekvenci oz. vsoti pojavitev besed oz. tipov entitet v posameznem članku. V drugem delu pa smo skupaj normalizirali še entitete in ključne besede, saj sta bili obe vrsti meta-podatkov predstavljeni z binarnimi značilkami.

Ko smo imeli na voljo reprezentacijo primerno za učenje, smo celotno množico označenih člankov naključno premešali in razdelili na dva dela, in sicer učno množico $\mathcal{U}$ in testno množico $\mathcal{T}$ v razmerju 80 : 20. Ker smo sprva poskušali poiskati optimalno kombinacijo značilk, parametrov in učnih algoritmov, smo nad $\mathcal{U}$ izvedli 5-kratno interno prečno preverjanje. Za preverjanje smo si izbrali šest kombinacij značilk: (1) unigami, (2) unigrami, entitete in ključne besede, (3) unigrami in frekvence NELL-ovih tipov entitet,

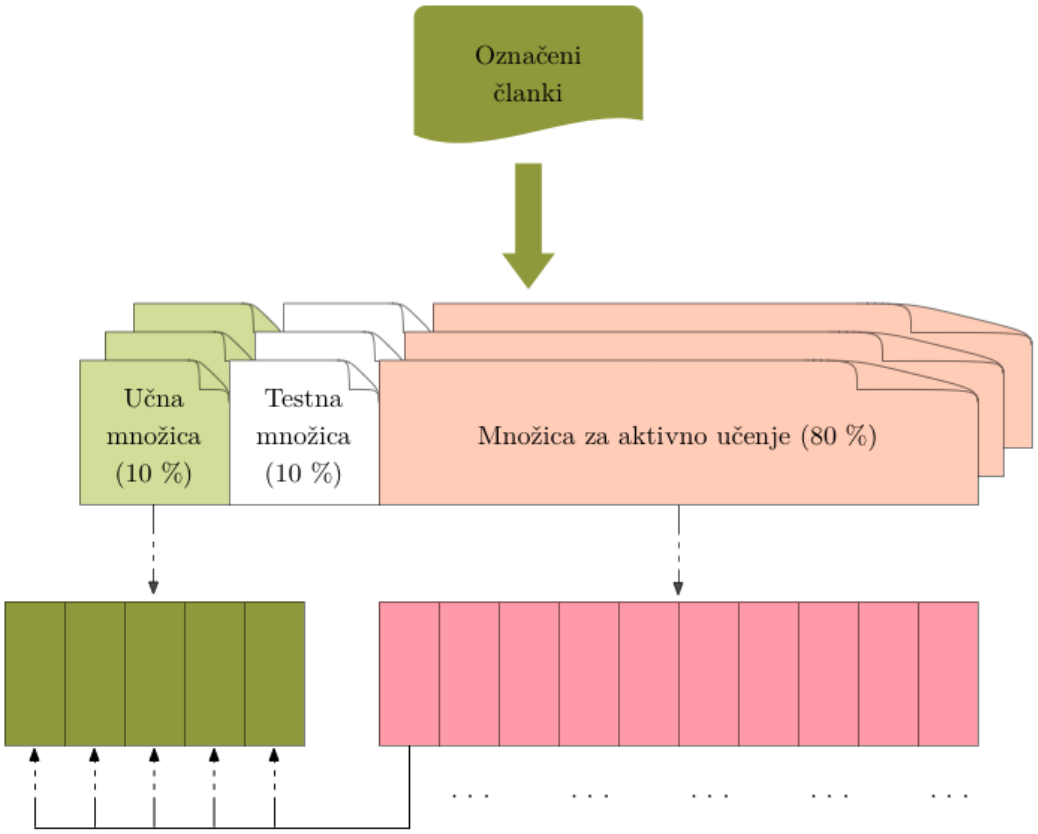
(4) unigrami in vsota verjetnosti NELL-ovih tipov entitet, (5) unigrami, entitete, ključne besede in frekvence NELL-ovih tipov entitet in (6) unigrami, entitete, ključne besede in vsota verjetnosti NELL-ovih tipov entitet. Slednje kombinacije so bile izbrane na podlagi zastavljene hipoteze, pri kateri se sprašujemo, ali značilke zgrajene iz NELL-ovih tipov entitet pomembenje vplivajo na izboljšanje modela kot drugi meta-podatki. Grobo rečeno, preverjati smo želeli kombinacije unigramov in meta-podatkov pridobljenih s pomočjo sistemov IJS ter preko spletnega programskega vmesnika The Guardian s kombinacijo unigramov in meta-podatkov pridobljenih iz baze znanja sistema NELL.

Celoten proces smo izvajali paralelno za štiri različne algoritme, in sicer logistično regresijo, SVM (z linearnim jedrom), naključne gozdove ter razdaljno uteženih k-najbližjih sosedov. Učenje vsakega od algoritmov smo izvedli s tremi različnimi parametri. Pri logistični regresiji in SVM smo spreminjali parameter $C = \frac{1}{\lambda}$, ki je inverzna vrednost regularizacijskega parametra λ. Za C smo uporabili vrednosti 1, 10 in 100. Pri naključnih gozdovih smo spreminjali število zgrajenih odločitvenih dreves, izbirali pa smo med 10, 50 in 100. Pri k-NN pa smo spreminjali parameter k, ki je imel zalogo vrednosti enako 11, 27 in 43. Vseh izbranih kombinacij je bilo torej $6 \times 4 \times 3 = 72$.

Po izvedbi 5-kratnega internega prečnega preverjanja smo glede na maksimalno vrednost F1-mere za vsako skupino atributov izbrali dve najbolj perspektivni kombinaciji parametrov. Z njimi smo nato nad celotno $\mathcal{U}$ zgradili dvanajst napovednih modelov in jih testirali na $\mathcal{T}$. S tem smo dobili končne rezultate v obliki F1-mere, ki nam povedo, kako dobro naj bi napovedni modeli klasificirali še ne videne podatke.

4.3 Pristopi k aktivnemu učenju

Pomemben del naloge je bila uporaba različnih pristopov za izbiranje primerov v okviru procesa aktivnega učenja in testiranje njihovih efektov. Uporabili smo štiri različne pristope za izbiranje primerov: (1) rob, (2) entropija,



Slika 4.3: Shema, ki prikazuje rokovanje s podatki pri procesu aktivnega učenja.

(3) kombinacija roba in korelacije ter (4) kombinacija entropije in korelacije. Obenem smo izbirali primere popolnoma naključno, s čimer smo želeli pridobiti minimalno referenčno vrednost.

Za lažjo predstavo rokovanja s podatki pri procesu aktivnega učenja, je na sliki 4.3 shematično prikazana delitev označenih primerov.

Učenje na množici vseh označenih primerov je zaradi znatne količine podatkov časovno zahtevno. Ta ugotovitev nas je privedla do postopka, po katerem smo si pri preverjanju različnih pristopov k aktivnemu učenju trikrat izbrali drugačno naključno podmnožico velikosti 10.000, ki je bila stratificirana in kot taka vsebovala 1.250 primerov vsakega razreda. Vsako od teh

podmnožic smo nadalje razdelili v razmerju $\mathcal{U} : \mathcal{T} : \mathcal{A} = 10 : 10 : 80$. $\mathcal{A}$ predstavlja množico primerov, ki bi bila v realnem primeru neoznačena in za katero bi morali izbrane primere posameznih pristopov še ročno označiti, preden bi jih lahko dodali v $\mathcal{U}$ in preverili njen vpliv na točnost klasifikatorja. V našem primeru pa temu ni bilo tako in so bili vsi primeri v njej označeni, saj je za namene takega poskusa bolj prikladno, da tekom procesa ne izvajaš ročnega označevanja. V fazi, kjer smo z različnimi pristopi izbirali primere iz množice $\mathcal{A}$, smo se pretvarjali, da primeri nimajo oznak oz. določenih kategorij, nato pa smo izbrane lahko dodali v $\mathcal{U}$ in takoj preverili efekte. S tem smo preskočili fazo ročnega označevanja. Za $\mathcal{U}$ ni značilno, da bi imela tako majhen delež, a smo se v tem primeru odločili za tako potezo, ker smo želeli, da osnovni klasifikator v 0. iteraciji ne dosega previsokih točnosti, da bi bili bolje vidni efekti sledečih iteracij.

Ker smo preverjali pet pristopov, smo petkrat reproducirali enako začetno $\mathcal{U}$, $\mathcal{T}$ pa je bila v danem izboru podmnožice velikosti 10.000 tekom vseh iteracij in za vse pristope, vedno enaka. Ker smo želeli izvesti deset iteracij izbire primerov in preverjanja efektov le-tega na točnost klasifikatorja, smo $\mathcal{A}$ naključno premešali in razdelili na deset delov, pri čimer smo prav tako pazili, da je bila porazdelitev razredov v vseh podmnožicah enakomerna. V vsaki od desetih podmnožic je bilo 800 člankov, v posamezni iteraciji pa smo vsakemu od petih pristopov dovolili, da izbere polovico (400) primerov. Tako je vsaka od petih $\mathcal{U}$ na začetku, v 0. iteraciji učenja, imela na voljo 1.000 primerov, proces aktivnega učenja pa je zaključila s 5.000 primeri. Da bi vnesli dodatno mero naključnosti, smo postopek naključnega mešanja $\mathcal{A}$ in izbire podmnožic ponovili desetkrat. Rezultate napovedovanja na podlagi desetih izbir različnih podmnožic smo za vsakega od pristopov povprečili po posameznih iteracijah. Za vsako od treh izbir podmnožic velikosti 10.000 smo tako dobili pet povprečnih krivulj (za vsak pristop eno), ki se spreminjajo skozi iteracije.

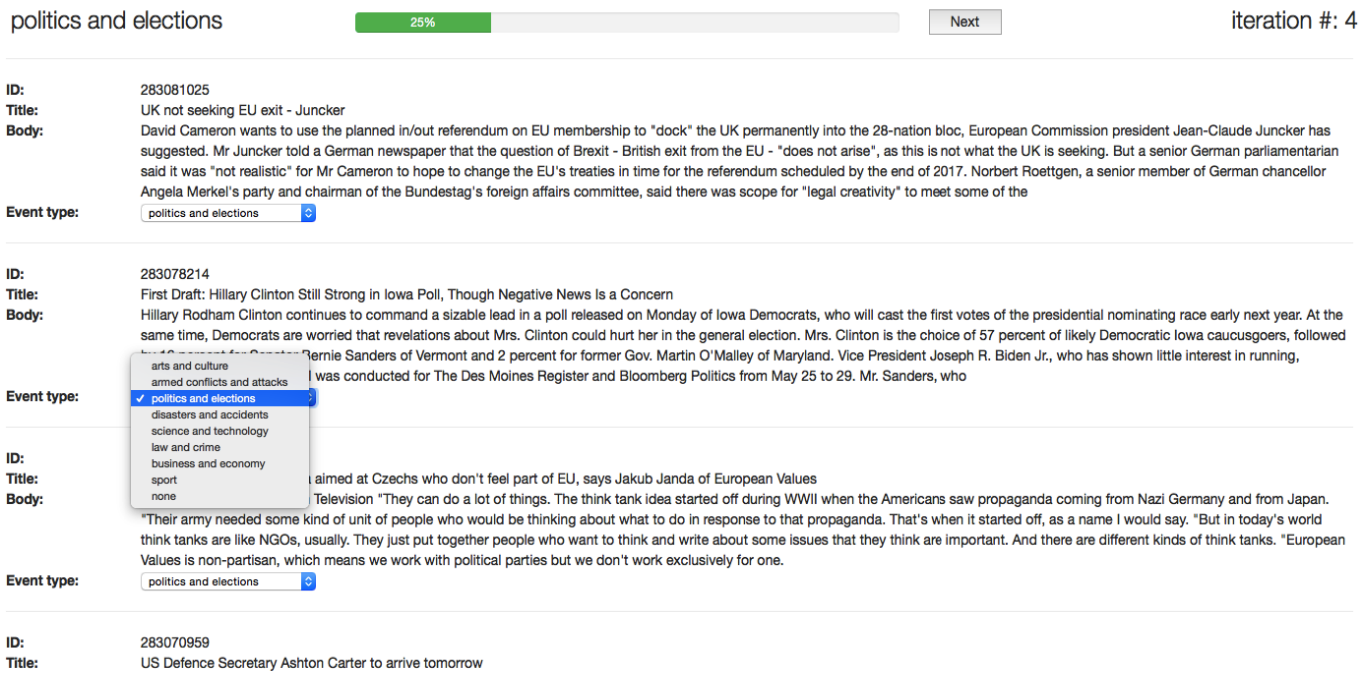
Hkrati smo pri obeh kombiniranih merah, na kateri vplivata negotovost in korelacija primerov z vsemi ostalimi primeri, preizkusil več vrednosti za

parameter γ. S tem smo spreminjali selekcijo primerov glede na vrednost njihove koreliranosti z ostalimi primeri.

4.4 Aktivno učenje na neoznačenih člankih

Da bi ugotovili, kako dobro naš klasifikator napoveduje kategorije za še ne videne neoznačene članke, smo iterativno izvajali proces aktivnega učenja. V vsaki iteraciji smo sprva napovedali oznake za podmnožico neoznačenih primerov velikosti okoli 3.000. Na podlagi izbranih pristopov k aktivnemu učenju smo izbrali 800 najbolj informativnih primerov (100 za vsako kategorijo v naši taksonomiji). Slednje smo izpostavili preko spletnega vmesnika za ročno preverjanje oznak, kjer so bili primeri za lažje označevanje ločeni v osem kategorij.

Slika 4.4 prikazuje spletni vmesnik za ročno določanje kategorij člankom, ki so bili izbrani kot najbolj informativni glede na različne pristope k aktivnemu učenju. Kot je razvidno na omenjeni sliki, je bilo člankom možno pripisati tudi oznako “none”, s čimer je ocenjevalec sporočil, da ima članek bodisi vsebinske napake (npr. da ni v angleškem jeziku ali pa da je vseboval neke druge oznake, ki ne bi smele biti del golega besedila članka) bodisi da ga ne more smiselno uvrstiti v nobeno od osmih kategorij.



Slika 4.4: Spletni vmesnik za ročno uvrščanje neoznačenih člankov v kategorije pri procesu aktivnega učenja (stran "politika in volitve").

Poglavje 5

Rezultati in diskusija

5.1 Klasifikacijski modeli

Pri gradnji klasifikacijskih modelov smo dobili zelo spodbudne rezultate. V tabeli 5.1 so prikazani rezultati 5-kratnega internega prečnega preverjanja na učni množici $\mathcal{U}$ velikosti 18.584 primerov. S tem postopkom smo želeli izbrati optimalno kombinacijo parametrov.

Skupine atributov oz. značilk so zaradi večje preglednosti podane s številkami od 1 do 6:

1 unigami

2 unigrami, entitete in ključne besede

3 unigrami in frekvence NELL-ovih tipov entitet

4 unigrami in vsota verjetnosti NELL-ovih tipov entitet

5 unigrami, entitete, ključne besede in frekvence NELL-ovih tipov entitet

6 unigrami, entitete, ključne besede in vsota verjetnosti NELL-ovih tipov entitet

Število značilk se je tekom gradnje modelov gibalo med približno 150.000 in 250.000, odvisno od količine meta-podatkov v $\mathcal{U}$.

Značilke	Algoritem	Parameter	Natančnost	Priklic	F1-mera
1	logistična regresija	1	0.8463	0.8447	0.8448
1	**logistična regresija**	**10**	**0.8522**	**0.8513**	**0.8514**
1	logistična regresija	100	0.8492	0.8484	0.8485
1	**SVM**	**1**	**0.8524**	**0.8513**	**0.8515**
1	SVM	10	0.8456	0.8447	0.8449
1	SVM	100	0.8425	0.8407	0.8411
1	naključni gozdovi	10	0.7243	0.7169	0.7161
1	naključni gozdovi	50	0.8059	0.7985	0.7983
1	naključni gozdovi	100	0.8200	0.8125	0.8123
1	k-NN	11	0.7765	0.3297	0.2991
1	k-NN	27	0.7981	0.7696	0.7715
1	k-NN	43	0.7991	0.7828	0.7831
2	logistična regresija	1	0.8577	0.8565	0.8567
2	**logistična regresija**	**10**	**0.8648**	**0.8643**	**0.8644**
2	logistična regresija	100	0.8635	0.8629	0.8630
2	**SVM**	**1**	**0.8643**	**0.8637**	**0.8638**
2	SVM	10	0.8608	0.8600	0.8602
2	SVM	100	0.8583	0.8569	0.8573
2	naključni gozdovi	10	0.7401	0.7329	0.7320
2	naključni gozdovi	50	0.8205	0.8131	0.8130
2	naključni gozdovi	100	0.8277	0.8199	0.8200
2	k-NN	11	0.7652	0.4711	0.4658
2	k-NN	27	0.7348	0.4617	0.4292
2	k-NN	43	0.7274	0.4286	0.3863
3	logistična regresija	1	0.8480	0.8465	0.8467
3	**logistična regresija**	**10**	**0.8549**	**0.8541**	**0.8542**
3	logistična regresija	100	0.8523	0.8516	0.8517
3	**SVM**	**1**	**0.8557**	**0.8547**	**0.8549**
3	SVM	10	0.8489	0.8483	0.8484

3	SVM	100	0.8458	0.8444	0.8447
3	naključni gozdovi	10	0.7293	0.7215	0.7204
3	naključni gozdovi	50	0.8088	0.8018	0.8015
3	naključni gozdovi	100	0.8226	0.8153	0.8151
3	k-NN	11	0.7920	0.4346	0.4448
3	k-NN	27	0.8021	0.7766	0.7785
3	k-NN	43	0.8044	0.7896	0.7899
4	logistična regresija	1	0.8480	0.8465	0.8466
4	**logistična regresija**	**10**	**0.8549**	**0.8541**	**0.8543**
4	logistična regresija	100	0.8523	0.8517	0.8518
4	**SVM**	**1**	**0.8557**	**0.8547**	**0.8549**
4	SVM	10	0.8487	0.8480	0.8482
4	SVM	100	0.8463	0.8450	0.8453
4	naključni gozdovi	10	0.7260	0.7190	0.7174
4	naključni gozdovi	50	0.8117	0.8049	0.8046
4	naključni gozdovi	100	0.8225	0.8163	0.8160
4	k-NN	11	0.7919	0.4345	0.4447
4	k-NN	27	0.8022	0.7767	0.7787
4	k-NN	43	0.8044	0.7895	0.7899
5	logistična regresija	1	0.8569	0.8558	0.8559
5	**logistična regresija**	**10**	**0.8644**	**0.8639**	**0.8640**
5	logistična regresija	100	0.8635	0.8631	0.8632
5	**SVM**	**1**	**0.8635**	**0.8629**	**0.8630**
5	SVM	10	0.8613	0.8606	0.8608
5	SVM	100	0.8583	0.8570	0.8574
5	naključni gozdovi	10	0.7416	0.7359	0.7349
5	naključni gozdovi	50	0.8142	0.8070	0.8069
5	naključni gozdovi	100	0.8255	0.8187	0.8186
5	k-NN	11	0.7680	0.4783	0.4742
5	k-NN	27	0.7369	0.4690	0.4390
5	k-NN	43	0.7310	0.4349	0.3949

6	logistična regresija	1	0.8570	0.8558	0.8560
6	**logistična regresija**	**10**	**0.8644**	**0.8639**	**0.8639**
6	logistična regresija	100	0.8634	0.8629	0.8630
6	**SVM**	**1**	**0.8635**	**0.8629**	**0.8631**
6	SVM	10	0.8612	0.8605	0.8607
6	SVM	100	0.8583	0.8570	0.8574
6	naključni gozdovi	10	0.7433	0.7358	0.7351
6	naključni gozdovi	50	0.8151	0.8078	0.8076
6	naključni gozdovi	100	0.8258	0.8194	0.8192
6	k-NN	11	0.7679	0.4782	0.4741
6	k-NN	27	0.7378	0.4688	0.4387
6	k-NN	43	0.7296	0.4346	0.3945

Tabela 5.1: Rezultati 5-kratnega internega prečnega preverjanja na učni množici označenih člankov.

Iz tabele 5.1 lahko razberemo, da v vseh skupinah atributov logistična regresija in SVM delujeta najbolje (gledano na podlagi F1-mere). Skupina 1, ki kot značilke vsebuje zgolj unigrame, daje pričakovano najnižjo vrednost F1-mere, kar pomeni, da meta-podatki nekoliko izboljšajo model. Najvišjo vrednost F1-mere dosegata omenjena algoritma pri skupini 2, kjer so unigramom dodani meta-podatki v obliki entitet in ključnih besed.

Pri naključnih gozdovih opazimo, da se z zviševanjem parametra, t.j. števila zgrajenih odločitvenih dreves, točnost napovedovanja izboljšuje. Če bi se želeli približati rezultatom algoritmov logistična regresija in SVM, bi morali znatno povečati ta parameter. Hkrati za naključne gozdove opazimo, da značilke, ki izvirajo iz meta-podatkov, ne pripomorejo bistveno k zvišanju vrednosti F1-mere pri zgrajenih modelih.

Metoda k-NN daje izmed vseh algoritmov najslabše rezultate. Pri njem je zanimivo, da najboljše rezultate dosega s skupino atributov 1, ki vsebuje le unigrame, kar očitno pomeni, da ga velika količina značilk zaradi dodatnih meta-podatkov, zgolj zmede pri napovedovanju.

Rezultati internega prečnega preverjanja so bili uporabljeni zgolj za izbiro optimalnih kombinacij značilk, algoritmov in parametrov. Končno napovedno točnost modelov pa je potrebno oceniti na ločeni testni množici $\mathcal{T}$, drugače so naše napovedi glede točnosti klasifikatorjev pristranske in s tem nepravilne. Ocenjevanje napovedne točnost na $\mathcal{T}$ nam torej pove, kako učinkovito bi napovedni modeli klasificirali še ne videne nove primere. Tako smo si na poglagi tabele 5.1 za vsako skupino atributov izbrali dve najboljši kombinaciji značilk, algoritmov in parametrov. V primeru vseh šestih skupin sta bili to kombinaciji (logistična regresija, $C = 10$) in (SVM, $C = 1$). S tema dvema kombinacijama smo nato na $\mathcal{U}$ zgradili dvanajst končnih napovednih modelov (za vsako skupino atributov dva modela), njihovo točnost pa smo preverili na $\mathcal{T}$.

Rezultati so predstavljeni v tabeli 5.2, iz katere lahko razberemo, da sta najvišjo vrednost F1-mere na $\mathcal{T}$ dosegla modela, ki sta bila zgrajena z algoritmom SVM pri skupinah atributov 5 in 6. Slednji skupini sta tisti, ki vsebujeta unigrame, entitete in ključne besede ter bodisi frekvence NELL-ovih tipov entitet bodisi vsote verjetnosti NELL-ovih tipov entitet. Ugotavljamo, da uporaba vseh meta-podatkov skupaj, pozitivno vpliva na končno vrednost F1-mere zgrajenih modelov, dočim pa razlika ni velika.

Če primerjamo skupine atributov 2, 3 in 4, ugotovimo, da kombinacija unigramov z meta-podatki pridobljenimi s pomočjo sistemov IJS in preko spletnega programskega vmesnika The Guardian, daje boljše rezultate kot kombinacija unigramov s tipi entitet pridobljenimi iz NELL-ove baze znanja.

V tabeli 5.3 so prikazani še rezultati napovedi za posamezne kategorije enega izmed dveh najboljših modelov (skupina atributov 6). S tabelo 5.4 pa prikažemo kontingečno matriko napovedi omenjenega modela.

Značilke	Algoritem	Parameter	Točnost	Priklic	F1-mera
1	logistična regresija	10	0.8583	0.8579	0.8579
1	SVM	1	0.8587	0.8579	0.8580
2	logistična regresija	10	0.8661	0.8657	0.8656
2	SVM	1	0.8693	0.8689	0.8688
3	logistična regresija	10	0.8583	0.8577	0.8577
3	SVM	1	0.8553	0.8545	0.8545
4	logistična regresija	10	0.8582	0.8577	0.8577
4	SVM	1	0.8553	0.8545	0.8545
5	logistična regresija	10	0.8659	0.8657	0.8655
5	**SVM**	**1**	**0.8704**	**0.8700**	**0.8699**
6	logistična regresija	10	0.8659	0.8657	0.8655
6	**SVM**	**1**	**0.8704**	**0.8700**	**0.8699**

Tabela 5.2: Rezultati testiranja modelov z optimalnimi kombinacijami značilk, parametrov in algoritmov na testni množici.

	Natančnost	Priklic	F1-mera
oboroženi spopadi in napadi	0.8678	0.8826	0.8751
umetnost in kultura	0.8906	0.8348	0.8618
posel in ekonomija	0.8695	0.8656	0.8675
katastrofe in nesreče	0.9117	0.8876	0.8995
zakon in kriminal	0.8158	0.7786	0.7968
politika in volitve	0.8039	0.8456	0.8243
znanost in tehnologija	0.9266	0.9368	0.9317
šport	0.9700	0.9791	0.9745
povprečje	0.8704	0.8700	0.8699

Tabela 5.3: Rezultati napovedi za posamezne kategorije najboljšega klasifikacijskega modela.

	oboroženi spopadi in napadi	**umetnost in kultura**	**posel in ekonomija**	**katastrofe in nesreče**	**zakon in kriminal**	**politika in volitve**	**znanost in tehnologija**	**šport**
oboroženi spopadi in napadi	**827**	5	3	23	36	59	0	0
umetnost in kultura	3	**293**	5	5	6	7	8	2
posel in ekonomija	4	2	**393**	1	21	26	4	1
katastrofe in nesreče	10	4	5	**537**	19	6	7	1
zakon in kriminal	34	9	13	20	**496**	33	0	3
politika in volitve	56	27	29	8	54	**734**	3	2
znanost in tehnologija	1	11	4	8	1	2	**341**	0
šport	2	0	2	3	4	1	1	**421**

Tabela 5.4: Kontingenčna matrika napovedi najboljšega klasifikacijskega modela.

	Natančnost	**Priklic**	**F1-mera**
Referenčni model	0.82	0.75	0.77
Naš model	**0.87**	**0.87**	**0.87**

Tabela 5.5: Primerjava referenčnega in našega modela.

Na podlagi tabele 5.3 in kontingenčne matrike 5.4 lahko ugotovimo, da klasifikator dosega najslabše napovedne rezultate za kategoriji "zakon in kriminal" ter "politika in volitve". Temu botruje dejstvo, da se ti dve kategoriji deloma prekrivata oz. da se članki iz ene kategorije velikokrat navezujejo na drugo kategorijo (npr. članki, ki govorijo o sprejemanju zakonodaje spadajo tako v prvo kot v drugo kategorijo, saj je govora o zakonih, hkrati pa gre za politični proces), kar pomeni, da ni popolnoma jasne ločnice med njima. Zaradi tega bi bilo smotrno zgraditi še klasifikator, ki ne napoveduje le enega, temveč več razredov, t.j. *večrazredni klasifikator* (angl. *multi-label classifier*). Po tem postopku bi torej lahko odkrili prekrivanja med kategorijami in kjer bi bilo potrebno, popravili obstoječo taksonomijo. Obenem pa bi lahko odkrili nove kategorije, s katerimi bi razširili taksonomijo, saj večrazredni klasifikatorji uvrščajo primere v več, enega ali nič razredov. Primere, ki ne bi bili uvrščeni v noben razred bi lahko s postopkom aktivnega učenja klasificirali v nove kategorije in na ta način razširili našo osnovno taksonomijo kategorij.

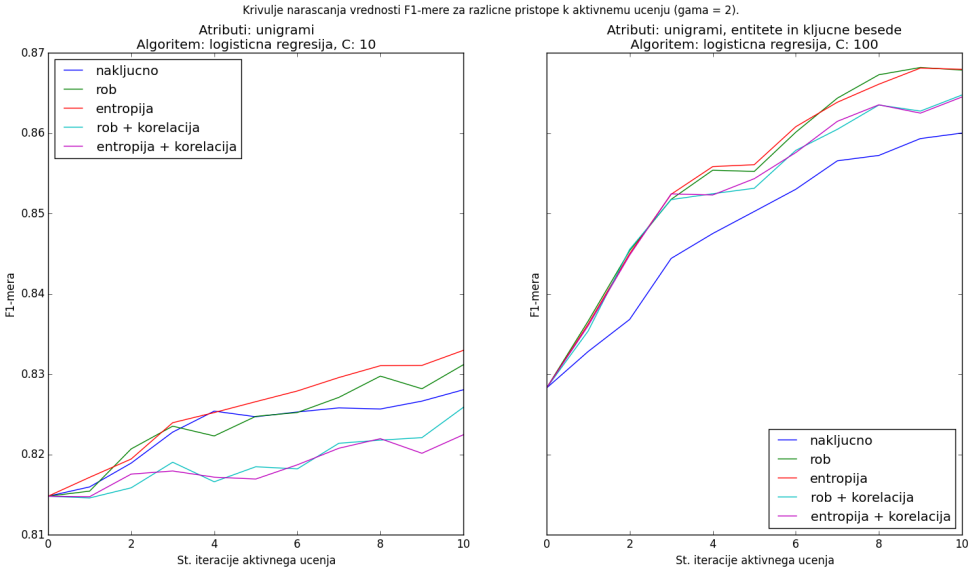
V tabeli 5.5 primerjamo rezultate referenčnega in našega najboljšega modela. Absolutno gledano smo referenčni model glede na F-1 mero izboljšali za približno 0,1 oz. 10 %, mu pri tem pomenljivo popravili priklic (povišan za 0,12 oz. 12 %) in natančnost (povišana za 0,05 oz. 5 %) ter hkrati ti dve meri praktično izenačili. Izboljšavo gre deloma pripisati povečani učni množici podatkov (okoli 4.000 primerov več), deloma pa obogatitvi modela s široko paleto različnih meta-podatkov (razvidno iz primerjave rezultatov skupin atributov 1 in 6). Vse to nakazuje, da so bili rezultati našega dela pozitivni.

5.2 Pristopi k aktivnemu učenju

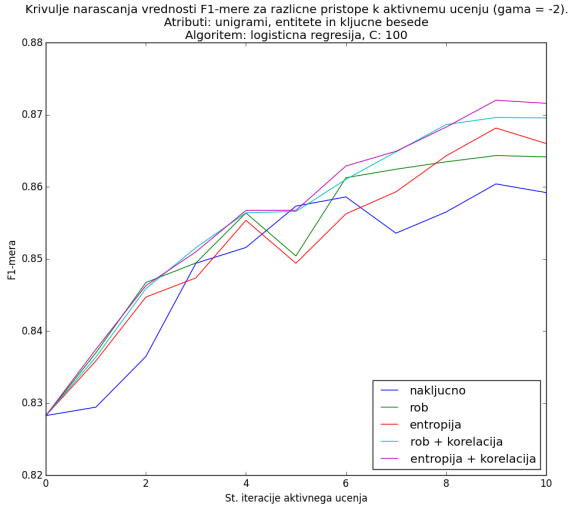
Na sliki 5.1 sta prikazani povprečni vrednosti F1-mere klasifikatorjev za pet različnih pristopov k aktivnemu učenju skozi deset iteracij v dveh izbirah podmnožic 10.000 člankov. Pri prvi izbiri podmnožice primerov smo imeli v skupini atributov zgolj unigrame, kot učni algoritem pa smo uporabili logistično regresijo s parametrom $C = 10$. Druga izbira podmnožice primerov je imela v skupini atributov poleg unigramov še entitete in ključne besede. Pri slednji izbiri smo kot učni algoritem prav tako uporabili logistično regresijo, a tokrat s parametrom $C = 100$. Obe kombinirani metriki sta imeli za parameter γ vrednost 2. Opazimo, da v primeru prve izbire podmnožice, naključno izbiranje primerov proizvede boljše rezultate kot izbiranje primerov na podlagi kombiniranih metrik, medtem ko je v primeru druge izbire podmnožice situacija ravno obratna. Torej lahko sklepamo, da dodatek metapodatkov v skupino atributov kombiniranima metrikama pomaga k boljšemu izbiranju primerov. Obenem opazimo, da sta najbolj perspektivna pristopa k aktivnemu učenju za izbiranje primerov glede na vrednost F1-mere rob in entropija, saj dosegata vidno boljše rezultate kot meri, ki sta kombinirani s korelacijsko metriko.

Na sliki 5.2 vidimo naraščanje vrednosti F1-mere za drugo izbiro podmnožice 10.000 primerov, kjer je bila kot učni algoritem uporabljena logistična regresija s parametrom $C = 100$. Tokrat smo za parameter γ izbrali vrednost -2. Opazimo, da sta vrednosti F1-mere za obe kombinirani metriki višji od vrednosti F1-mere metrik, ki izbirata primere zgolj na podlagi negotovosti napovedi klasifikatorja. Na podlagi rezultatov prikazanih na slikah 5.1 in 5.2 sklepamo, da je v primeru, ko damo večji pomen primerom, ki imajo vrednost korelacijske metrike na spodnjem delu intervala med 0 in 1, vpliv na vrednosti F1-mere boljši, kot v primeru, ko je večji pomen dan primerom, ki imajo vrednosti korelacijske metrike na zgornjem delu omenjenega intervala.

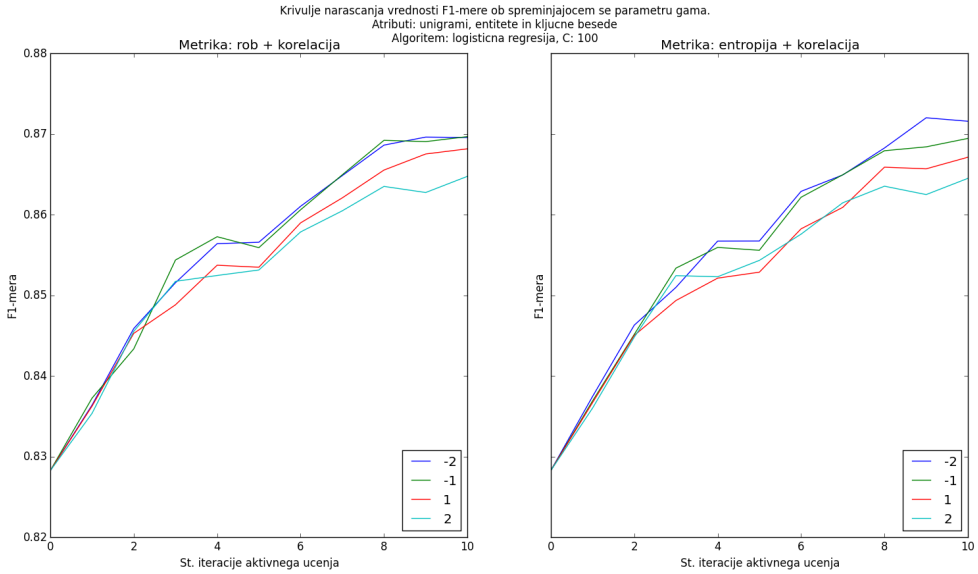
Za boljšo predstavo vpliva parametra γ na vrednosti F1-mere si poglejmo sliko 5.3. Pri tem so na levi strani prikazane krivulje naraščanja vrednosti



Slika 5.1: Krivulje naraščanja vrednosti F1-mere različnih pristopov k aktivnemu učenju za prvi dve izbiri podmnožic 10.000 primerov, pri čimer je vrednost parametra γ enaka 2.



Slika 5.2: Krivulje naraščanja vrednosti F1-mere različnih pristopov k aktivnemu učenju za drugo izbiro podmnožice 10.000 primerov, pri čimer je vrednost parametra γ enaka -2.



Slika 5.3: Krivulje naraščanja vrednosti F1-mere v primeru kombiniranih metrik za drugo izbiro podmnožice 10.000 primerov ob spreminjajočem se parametru γ.

F1-mere za kombinirano metriko roba in korelacije, na desni pa za kombinirano metriko entropije in korelacije. Iz slike 5.3 je ponovno razvidno, da negativne vrednosti za parameter γ, ki pripišejo večji pomen primerom, ki imajo vrednosti korelacijske metrike na spodnjem delu intervala med 0 in 1, dajejo višje vrednosti F1-mere.

Izbiro primerov na podlagi kombiniranih metrik bi se dalo izboljšati z gručenjem primerov, s čimer bi dobili dodatno informacijo o tem, na kakšen način je določen primer povezan z ostalimi.

Ker naš cilj pri preverjanju učinkov različnih pristopov k aktivnemu učenju ni bil maksimizacija točnosti napovednega modela, moramo rezultate na slikah 5.1, 5.2 in 5.3 upoštevati relativno in ne absolutno. Hkrati je to tudi razlog, da smo se pri testiranjih odločili za uporabo algoritma logistične regresije in ne SVM (prvi deluje veliko hitreje kot drugi).

5.3 Aktivno učenje na neoznačenih člankih

Pri aktivnem učenju na podlagi neoznačenih člankov, kjer smo preko spletnega vmesnika izvajali postopek ročnega označevanja, je bilo ugotovljeno, da klasifikator povečini deluje dobro za kategorije "oboroženi spopadi in napadi", "posel in ekonomija", "zakon in kriminal", "politika in volitve", "znanost in tehnologija" in "šport", dela pa določene napačne napovedi pri uvrščanju v kategoriji "umetnost in kultura" in "katastrofe in nesreče".

V kategorijo "umetnost in kultura" klasifikator velikokrat uvrsti članke, ki govorijo o televizijskih, filmskih in glasbenih zvezdnikih ter ostalih estradnikih, kar kaže na pomanjkanje ustrezne kategorije, ki bi zajela članke s to vsebino. Podobno težavo ima klasifikator pri kategoriji "katastrofe in nesreče", kamor uvršča članke, ki govorijo o vremenu ali pa predstavljajo preproste vremenske napovedi. Na podlagi opisanih ugotovitev ocenjujemo, da bi bilo obstoječo taksonomijo potrebno razširiti s kategorijama, ki bi bili namenjeni člankom o zvezdnikih in člankom o vremenu.

Poglavje 6

Sklepne ugotovitve

Namen diplomskega dela je bila izgradnja klasifikatorja, ki bi znal kar se da dobro uvrščati novičarske članke v predhodno opredeljene kategorije. Gradili smo na podlagi referenčnega modela, ki smo ga skušali izboljšati [8]. S pridobitvijo kakovostnih označenih člankov s portala dogodkov Wikipedia in s pomočjo spletnega programskega vmesnika The Guardian ter dodatnih meta-podatkov iz baze znanja sistema NELL, nam je to v veliki meri tudi uspelo, saj smo vrednost F1-mere glede na referenčni model izboljšali za 10 %, obenem pa za 5 % povečali natančnost in za 12 % popravili priklic. Ena od glavnih pomanjkljivosti referenčnega modela je bil nizek priklic, ki smo ga z našim modelom uspeli občutno izboljšati. Pri gradnji klasifikatorjev smo hkrati ugotovili, da preprosti algoritmi kot k-NN zelo slabo delujejo na kompleksnih problemih z velikim številom značilk, medtem ko je algoritem SVM najbolj primeren za take probleme, saj zna modelirati kompleksne podatke in pri procesu učenja hipoteze uporabiti številne, tudi manj pomembne značilke.

Prav tako nas je zanimalo, če dodatek meta-podatkov pridobljenih iz baze znanja sistema NELL pozitivno vpliva na rezultate napovedi modela. Izkazalo se je, da je pozitiven vpliv razviden le, če so omenjeni meta-podatki uporabljeni skupaj z entitetami pridobljenimi s pomočjo sistema Enrycher ter ključnimi besedami pridobljenimi preko sistema Event Registry in spletnega

programskega vmesnika The Guardian. Zanimiva potencialna izboljšava modela bi bila uporaba relacij, ki jih za par entitet hrani baza znanja sistema NELL.

Obenem smo preizkusili pet različnih pristopov za izbiro primerov pri procesu aktivnega učenja. Pokazali smo, da vse štiri informirane metode delujejo bolje kot naključno izbiranje primerov. Za najboljše so se izkazale kombinirane metrike, ki mero negotovosti napovedi kombinirajo s koreliranostjo med primeri, v primeru, da je bila vrednost pripadajočega parametra γ negativna. Delovanje metod, ki na podlagi posteriornih verjetnosti napovedi klasifikatorja računajo negotovost, je bilo primerljivo s kombiniranimi metrikami, a vendar za malenkost slabše. Na podlagi tega zaključujemo, da je informirano aktivno učenje izjemno dobra metoda za pridobitev novih kakovostnih označenih primerov in hkrati dodajamo, da bi se učinkovitost kombiniranih metod, ki delujejo na podlagi negotovosti ter koreliranosti med primeri dalo izbojšati z uporabo nenadzorovanega učenja, natančneje algoritmov za gručenje, ki bi ponudili nov pogled na informativnost opazovanih primerov.

Na podlagi ročnega uvrščanja neoznačenih člankov med procesom aktivnega učenja je bila ugotovljena potreba po razširitvi osnovne taksonomije osmih kategorij. Predlagani novi kategoriji bi vsebovali članke o zvezdnikih in estradnikih ter vremenu. Zanimiv nov pristop k danemu klasifikacijskemu problemu bi lahko vključeval večrazredno klasifikacijo, pri kateri lahko posamezni članek namesto v eno kategorijo uvrstimo v več kategorij hkrati. S tem bi avtomatsko identificirali prekrivajoče se in manjkajoče kategorije in posledično zelo poenostavili redefinicijo začetne taksonomije.

V skladu z dobrimi rezultati napovedi zgrajenih modelov v primeru enonivojske taksonomije ugotavljamo, da bi se dalo enak postopek uporabiti pri gradnji klasifikatorjev, ki bi skušali napovedovati bolj specializirane kategorije na več nivojih hierarhične taksonomije. Pozitivni rezultati preizkušanja pristopov k aktivnemu učenju nakazujejo na dejstvo, da bi bilo po krajši začetni fazi neinformiranega ročnega uvrščanja v nove specifične kategorije, za

nadaljnje pridobivanje potrebne količine označenih primerov, ki bi jih potrebovali za učenje modelov na različnih nivojih hierarhije, smiselno uporabiti enega od omenjenih pristopov.

Dodatek A

Programska koda

Iskanje optimalne kombinacije značilk, parametrov in učnih algoritmov s pomočjo internega prečnega preverjanja:

```
from Data import Features
from Util import Classifier

# load training set features
features_train = Features(
    dir_name='wiki_guardian/train',
    labeled=True)
features_train.load()

# define combinations for parameter optimization grid search
feature_indices = [0, 1, 2, 3, 4, 5]
combinations = {'knn': [11, 27, 43], 'rf': [10, 50, 100],
                'svm': [1, 10, 100], 'logreg': [1, 10, 100]}
```

```
# perform grid search using k-fold cross-validation with
# different feature, algorithm and parameter combinations
clf = Classifier(
    mode='cv',
    k=5,
    feature_indices=feature_indices,
    combinations=combinations)
clf.fit_predict(train=features_train)
clf.evaluate_and_save_results()
```

Postopek delitve množice označenih primerov in preizkušanje različnih pristopov k aktivnemu učenju:

```
import numpy as np
from Data import WikiGuardian, Features
from Util import NELLDict, ActiveLearning

seed1 = 1811
seed2 = 1103

# load articles
wg = WikiGuardian()
wg.load('articles_preprocessed.json')

# shuffle article ids of the entire article set
article_ids = np.array(wg.article_dict.keys())
order = np.arange(len(article_ids))
np.random.seed(seed1)
np.random.shuffle(order)
article_ids = article_ids[order]

# select stratified subset (small no. of articles per event type)
selected = wg.select_stratified_subset(
    shuffled_article_ids=article_ids,
    n_per_event_type=1250)
article_ids = np.array(sum(selected.values(), []))
order = np.arange(len(article_ids))
np.random.shuffle(order)
article_ids = article_ids[order]
```

```
# divide article ids
n_train = int(round(len(article_ids) * 0.1))
n_test = int(round(len(article_ids) * 0.1))
train_ids = article_ids[:n_train]
test_ids = article_ids[n_train:n_train+n_test]
active_learning_ids = article_ids[n_train+n_test:]

# shuffle active learning set ids
order = np.arange(len(active_learning_ids))
np.random.seed(seed2)
np.random.shuffle(order)
active_learning_ids = active_learning_ids[order]
n_inner = 10
n_per_iter = len(active_learning_ids) / n_inner

approaches = ['random', 'margin', 'entropy',
              'margin-correlation', 'entropy-correlation']

# split articles into 3 subsets and save them
for app in approaches:
    wg.select_subset(train_ids, 'split/%s/train_0.json' % app,
                     save=True)
for n in range(n_inner):
    wg.select_subset(
        active_learning_ids[n*n_per_iter:(n+1)*n_per_iter],
        'split/active_learning_%d.json' % n, save=True)
wg.select_subset(test_ids, 'split/test.json', save=True)

# load NELL dict
nell = NELLDict()
nell.load()
```

```
# load initial training sets and create features
for app in approaches:
    wg.load('split/%s/train_0.json' % app)
    features_train = Features(
        dir_name='wiki_guardian/%s/train_0' % app,
        labeled=True)
    features_train.transform(wg.article_dict,
                             nell.entity_categories)

# load active learning sets and create features
for n in range(n_inner):
    wg.load('split/active_learning_%d.json' % n)
    features_active_learning = Features(
        dir_name='wiki_guardian/active_learning_%d' % n,
        labeled=True)
    features_active_learning.transform(wg.article_dict,
                                       nell.entity_categories)

# load test set and create features
wg.load('split/test.json')
features_test = Features(
    dir_name='wiki_guardian/test',
    labeled=True)
features_test.transform(wg.article_dict,
                        nell.entity_categories)

# number of articles chosen per approach per iteration
n_subset = 400
```

```
for n in range(n_inner):
    # load active learning set articles and features
    wg_active_learning = WikiGuardian()
    wg_active_learning.load('split/active_learning_%d.json' % n)
    features_active_learning = Features(
        dir_name='wiki_guardian/active_learning_%d' % n,
        labeled=True)
    features_active_learning.load()

    for app in approaches:
        # load training set articles and features
        wg_train = WikiGuardian()
        wg_train.load('split/%s/train_%d.json' % (app, n))
        features_train = Features(
            dir_name='wiki_guardian/%s/train_%d' % (app, n),
            labeled=True)
        features_train.load()

        # select a subset of articles according to the active
        # learning instance selection approach
        al = ActiveLearning(
            n=n, approach=app, n_subset=n_subset,
            features_train=features_train,
            features_active_learning=features_active_learning,
            feature_indices=[1],
            combinations={'logreg': [100]},
            seed=seed, beta=-2)
        article_ids = al.select()
```

```
# add the subset to the appropriate training set
article_dict = wg_active_learning.select_subset(
    article_ids, save=False)
wg_train.join(
    [wg_train.article_dict, article_dict],
    'split/%s/train_%d.json' % (app, n+1))

# create new training set features
features_train = Features(
    dir_name='wiki_guardian/%s/train_%d' % (app, n+1),
    labeled=True)
features_train.transform(wg_train.article_dict,
                         nell.entity_categories)
```

Razred, ki predstavlja ogrodje za gradnjo slovarja tipov entitet sistema NELL:

```
from collections import defaultdict
import requests
import json
from mllib import DataLoader, DataSaver

class NELLDict:
    def __init__(self, path='data/nell/entity_categories.json'):
        self.path = path
        self.NELL_API_URL = 'http://rtw.ml.cmu.edu/rtw/api/json0'
        self.QUERY_SUCCESS = 'NELLQueryDemoJSON0'
        self.entity_categories = defaultdict(list)

    class _Category:
        def __init__(self):
            self.name = ''
            self.score = 0.0

    def update(self, article_dict):
        # query NELL API and add new entity categories to dict
        for article_id, article in article_dict.iteritems():
            for entity in article.entities + article.keywords:
                if entity not in self.entity_categories:
                    self._query(entity)

        self._save()
```

```
def _query(self, entity):
    # HTTP request & response
    try:
        params = dict(lit1=entity.replace(' ', '_'),
                      predicate='*')
        response = requests.get(self.NELL_API_URL, params)
        data = json.loads(response.text)
    except requests.exceptions.ConnectionError:
        return

    # success
    if data['kind'] == self.QUERY_SUCCESS:
        # no categories returned
        if not data['items']:
            return

        # categories found, parse the response and store it
        else:
            for item in data['items']:
                cat = self._Category()
                cat.name = item['predicate'].encode('utf-8')
                cat.score = float(
                    item['justifications'][0]['score'])
                self.entity_categories[entity].append(cat)

    # error
    else:
        print data['message']
        return
```

```
    def _save(self):
        DataSaver.save(
            self.path,
            {e: [c.__dict__ for c in cat]
             for e, cat in self.entity_categories.iteritems()})

    def load(self):
        # load existing NELL entity categories
        entity_categories_dict = DataLoader.load(self.path)

        for entity, categories in \
            entity_categories_dict.iteritems():
            for category in categories:
                cat = self._Category()
                cat.name = category['name']
                cat.score = category['score']
                self.entity_categories[entity].append(cat)
```

Literatura

[1] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[2] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr. in T. M. Mitchell, "Toward an Architecture for Never-Ending Language Learning", v *Proceedings of the 24th Conference on Artificial Intelligence* (AAAI '10), str. 8, 2010.

[3] N. A. Diamantidis, D. Karlis in E. A. Giakoumakis, "Unsupervised stratification of cross-validation for accuracy estimation", *Artificial Intelligence*, št. 116, zv. 1–2, str. 1–16, 2000.

[4] Y. Fu, X. Zhu in B. Li, "A survey on instance selection for active learning", *Knowledge and Information Systems*, št. 35, zv. 2, str. 249–283, 2012.

[5] M. Ikonomakis, S. Kotsiantis in V. Tampakas, Text classification using machine learning techniques, *World Scientific and Engineering Academy and Society Transactions on Computers*, št. 4, zv. 8, str. 966–974, 2005.

[6] I. Kononenko in M. R. Šikonja. *Inteligentni sistemi.* Založba FE in FRI, 2010.

[7] I. Kononenko in M. Kukar. *Machine Learning and Data Mining: Introduction to Principles and Algorithms.* Horwood Publishing Limited, 2007.

[8] A. Košmerlj, E. Belyaeva, G. Leban, B. Fortuna in M. Grobelnik, "Towards a complete event type taxonomy", v *Proceedings of the 24th International Conference on World Wide Web Companion* (WWW '15), str. 899–902, 2015.

[9] G. Leban, B. Fortuna, J. Brank in M. Grobelnik, "Cross-lingual detection of world events from news articles", v *Proceedings of the 13th International Semantic Web Conference* (ISWC '14), str. 21–24, 2014.

[10] G. Leban, B. Fortuna, J. Brank in M. Grobelnik, "Event registry: learning about world events from news", v *Proceedings of the 24th International Conference on World Wide Web Companion* (WWW '14), str. 107–110, 2014.

[11] J. Leskovec, A. Rajaraman in J. D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2014.

[12] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson idr., "Never-Ending Learning", v *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (AAAI '15), str. 9, 2015.

[13] M. C. Monard, G. E. A. P. A. Batista in J. M. Abe, "Learning with skewed class distributions", *Advances in Logic, Artificial Intelligence and Robotics*, št. 85, str. 173–180, 2002.

[14] R. C. Prati, M. C. Monard, G. E. A. P. A. Batista in J. M. Abe, "Learning with class skews and small disjuncts", v *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence* (SBIA '04), str. 296–306, 2004.

[15] Programska knjižnica matplotlib. [Online]. Dosegljivo: http://matplotlib.org. [Dostopano 2.9.2015].

[16] Programska knjižnica NumPy. [Online]. Dosegljivo: http://www.numpy.org. [Dostopano 2.9.2015].

[17] Programska knjižnica scikit-learn. [Online]. Dosegljivo: http://scikit-learn.org/stable/. [Dostopano 2.9.2015].

[18] Programska knjižnica SciPy. [Online]. Dosegljivo: http://www.scipy.org. [Dostopano 2.9.2015].

[19] Programska knjižnica scrapy. [Online]. Dosegljivo: http://scrapy.org. [Dostopano 2.9.2015].

[20] Programski jezik JavaScript. [Online]. Dosegljivo: https://developer.mozilla.org/en/JavaScript. [Dostopano 2.9.2015].

[21] Programski jezik jQuery. [Online]. Dosegljivo: https://jquery.com. [Dostopano 2.9.2015].

[22] Programski jezik PHP. [Online]. Dosegljivo: http://php.net. [Dostopano 2.9.2015].

[23] Programski jezik Python. [Online] Dosegljivo: https://www.python.org [Dostopano 2. 9. 2015].

[24] Programski paket za krnjenje besedil Porter Stemmer knjižnice NLTK. [Online]. Dosegljivo: http://www.nltk.org/_modules/nltk/stem/porter.html. [Dostopano 2.9.2015].

[25] S. J. Russell in P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall series in artificial intelligence. Prentice Hall, 2010.

[26] Shema delovanja sistema Enrycher. [Online] Dosegljivo: http://ailab.ijs.si/wp-content/uploads/2011/07/Dependencies_Scale1.png [Dostopano 2. 9. 2015].

[27] T. Štajner in D. Mladenić, "Entity resolution from texts using statistical learning and ontologies", v *Proceedings of the 4th Asian Conference on The Semantic Web* (ASWC '09), str. 91–104, 2009.

[28] T. Štajner, D. Rusu, L. Dali, B. Fortuna, D. Mladenić in M. Grobelnik, "A service oriented framework for natural language text enrichment", *Informatica (Slovenia)*, št. 34, zv. 3, str. 307–313, 2010.

[29] S. Tong in D. Koller, "Support vector machine active learning with applications to text classification", *The Journal of Machine Learning Research*, št. 2, str. 45–66, 2002.

[30] M. Trampuš in B. Novak, "The internals of an aggregated web news feed", v *Proceedings of the 15th International Multiconference on Information Society* (IS '12), str. 4, 2012.